Master's Thesis - Econometrics & Management Sciences:
Business Analytics & Quantitative Marketing

Faster Simulation for Colon-cancer Screening: a Meta-modeling approach

Arne E. Barkema
(473291)

Supervisor:	dr. S. (Shadi) Sharif Azadeh
Second assessor:	dr. K.S. (Krzysztof) Postek
External Supervisors:	dr. I. (Iris) Lansdorp-Vogelaar
	A. (Andrea) Gini, MSc.

February, 2020

Abstract

Colorectal Cancer poses a heavy burden on the population, which can be relieved through screening programmes. The cost-effectiveness of different screening programmes can be predicted through simulation software, such as MISCAN-Colon. However, running the software is often too computationally expensive to evaluate all scenarios of interest, which may leave optimal screening programmes unexplored. This study compares different strategies to develop a metamodel for MISCAN-Colon, which can assist in the faster evaluation of screening programmes. Towards that purpose, it compares different strategies for sampling data and different model architectures.

Based on a simulation study of MISCAN-Colon, five sampling strategies, which vary in how they sample risk and screening, and 168 Neural Network architectures are proposed. All model architectures are evaluated for all sampling strategies, which yields an optimal architecture per sampling strategy. For each sampling strategy a Bayesian Neural Network is fitted to function as a metamodel, which is then used to make predictions on a universal test set. The performance of each metamodel is then evaluated in terms of speed, accuracy and uncertainty estimation. The results indicate that randomly sampling screening and sampling risk from the population distribution yields the most accurate metamodel, although more work is needed to make it usable in practice.

Acknowledgements

I would like to express my gratitude to Martijn Starmans and Florian Dubost of Biomedical Imaging Group Rotterdam (BIGR) for offering guidance in Machine Learning matters. Their guidance allowed for a better targeted use of models and to an altogether more refined product. Furthermore, generating the large amount of data used in this thesis was made possible through the Bebop computer of the Argonne National Laboratory, which employs the Extreme-scale Model Exploration with Swift (EMEWS) framework. This was made possible through Jonathan Ozick and and Nicholson Collier, as part of the CISNET collaboration.

Contents

Acronyms

ANN Artificial Neural Network. 8, 10, 12–14, 50, 51, 63, 64

BN Batch Normalization. 13

BNN Bayesian Neural Network. 14, 63

DGP Deep Gaussian Process. 63

KL Divergence Kullback-Leibler Divergence. 63

LYG Life Years Gained. 7

MCD Monte Carlo Dropout. 63

MSE Mean Squared Error. 63

RBF Radial Basis Functions. 8, 50

RF Random Forest. 8, 50

SVM Support-Vector Machines. 8, 50

1 Introduction

Globally, Colorectal Cancer (CRC) is the fourth most common cancer and the fifth most frequent cause of cancer death (Bray et al., 2018). Correlated with lifestyle factors that accompany growing prosperity, such as obesity, CRC is a growing problem in the developed world.

Although global incidence has sharply risen between 1990 (818,000 diagnoses) and 2013 (1.6 million diagnoses), mortality rates have decreased due to better treatment methods and the use of screening programmes. In cancer screening programmes, a certain portion of the population is invited to undergo some cancer detection test. Effectiveness and cost-effectivenss of such programmes have been demonstrated in the medical literature (Lauby-Secretan, Vilahur, Bianchini, Guha, & Straif, 2018).

Screening programmes are already successfully being implemented in multiple continents. The (Dutch) National Institute for Public Health and the Environment (RIVM) conducts population studies on the prevalence of cancer in its citizens. Towards this purpose, they direct a scheme that invites Dutch citizens to partake in a Fecal Immunochemical Test (FIT), which can be self-administered at home and can be used to diagnose (with a certain accuracy) CRC. These citizens are invited to participate every two years, starting at the age of 55, until the age of 75 (Toes-Zoutendijk et al., 2017). Although not implemented, the medical societies in the United States have declared the importance of CRC screening. The US Preventive Services Task Force (USPSTF) recommends starting screening at age 50 and the American Cancer Society (ACS) recommends starting at age 45.

There are some studies that indicate that this is an effective regime for the timely detection of (pre)cancereous lesions. However, given the current available national resources (Van Hees et al., 2015), there is no evidence that this is optimal for all individuals in the population. Some individuals may be at higher risk of developing CRC due to several genetic or environmental (lifestyles) factors. Screening should only be used where its impact is the greatest. By focusing screening on the right individuals at the right time, governments or policy makers implement screening policies that cost-effectively optimize life-years for the population, minimizing cancer-related suffering in high-risk individuals and unnecessary screening intervention in low-risk individuals.

Already much is known about which personal characteristics are associated with an increased risk of colon cancer (Ma & Ladabaum, 2014; Usher-Smith et al., 2018). Considering the results of many studies conducted to quantify the individual risk of developing CRC based on lifestyle factors (Johnson et al., 2013), genetic indicators and comorbidity (Haggar & Boushey, 2009), a possible way to reduce 'unnecessary' screenings may be to personalize screening based on individual CRC risk. However, this information is not yet used in the design of nation-wide screening programmes. Personalized CRC screening is, therefore, one of the challenges that health organizations need to face in the near future.

Because real-life implementation of screening is costly and takes a long time to return results, the cost-effectiveness of screening is often estimated through simulation studies. Well-known simulation models for CRC, such as MISCAN-Colon (Loeve, Boer, van Oortmarssen, van Ballegooijen, & Habbema, 1999), are used to provide estimates of costs and benefits of implementing specific screening programmes, for simulated populations. The strategic scheduling of screening tests may be seen as a stochastic optimization problem, in which costs are minimized and (quality) life-years are maximized. Unfortunately, evaluating these scenarios, even through simulation, is expensive, which slows down progress in this field of research.

Several studies (e.g. Segnan et al. (2007); Singh, Turner, Xue, Targownik, and Bernstein (2006)) provide evidence for effective screening intervals when applying a single test, but no study yet has found a cost-effective regime that combined different screening measures. Most notably Naber (2017) has stratified a population based only on optimal colonoscopy frequency for different risk groups in the US. Again, researchers are limited by the long run time of the simulation model they work with. This limits the amount of scenarios they could simulate and therefore the amount of solutions they could explore.

Both risk stratification and multiple testing are difficult to optimize, due to the cost of running simulations. In many scientific disciplines, the burden of costly simulation models is alleviated by cheaper approximations of the model dynamics, so-called metamodels (also: surrogate models, response-surface models). These metamodels are trained on the main model's input and output and are designed to emulate their approximate relationship. Using these metamodels, scientists are able to evaluate many more scenarios within the same time, with approximate certainty, and explore relevant regions of the solution space. Cost-effective solutions obtained by the metamodels can then be verified by the main model. Machine Learning frameworks are often used as metamodels.

The main aim of this thesis is to deliver a metamodel design that researchers can use to evaluate screening strategies in a fraction of the time it takes the original model to run. Towards that purpose, we study optimal sampling methods, frameworks, designs and evaluation methods for metamodels of expensive simulation models. This metamodel is able to make inferences on costs and benefits, based on screening and cancer risk inputs. Such a model should allow for risk-based stratification and run many times faster than the main model. In its estimates, accuracy should be balanced throughout the input range, not favor, e.g. certain risk ranges. Furthermore, to validate its estimates, the metamodel should not only predict the mean of predictive distribution, but also the uncertainty.

The findings and algorithms included in this thesis have important implications, allowing researchers and policy-makers to find optimal screening trajectories at individual level, to stratify the population, and to determine optimal screening programmes for citizens

based on their individual risk, at greater speed than before.

This thesis starts by providing the background knowledge required to understand the methods used in this thesis, which is section 2. Then follows the Methods section, section 3, in which data sampling methods are discussed and different experiments are laid out that are required to develop the metamodel. After that, the results are presented in section 4. Finally, we reflect on the results in section 5. To investigate the effect of alternative initial settings, a sensitivity analysis is provided in section 6.

2 Background

This section provides supplementary information the reader may require to understand the methods that are used in this thesis. This section starts with a short introduction of the natural history of CRC and the relevance of screening, section 2.1. Then, the workings and dynamics of the main model, MISCAN-Colon are explained in section 2.2. After that, an analysis of the uncertainty dynamics of the main model is provided, section 2.3. The subsequent section focuses on metamodels, their core functioning, different types and how they can model uncertainty, section 2.4. The next section discusses different sampling strategies for metamodel training, section 2.5. After that, a section will focus on Neural Networks specifically, section 2.6. Finally, an overview follows of uncertainty in neural networks, section 2.7.

2.1 CRC Development and Intervention

Colorectal Cancer is a malignant neoplasm in the colon or rectum. CRC starts as a non-malignant polyp and may progress through several stages, including death from CRC. CRC is commonly the result of a long process (Ristvedt et al., 2005), starting with the onset of adenomas (precancerous lesions) that may slowly progress and become malignant cancers (Morson, 1974; Vogelstein et al., 1988). CRC screening programmes have the potential to detect precancerous lesions, after which they will be removed (preventing, therefore, the further development of cancer), even before it comes symptomatic. Earlier detection of the disease will make a patient better treatable. The 5-year survival rate is 90% for early detection, but only 10% for people diagnosed with distant metastatic CRC (Jemal et al., 2004; Ries et al., 2006). Nevertheless, without centralized screening, patients will often not seek testing or treatment, even when CRC may become symptomatic. For example, rectal bleeding is often not recognized by patients as CRC, which delays diagnosis and, consequently, treatment. Thus, screening is one of the most important and effective tools to reduce CRC mortality.

There are two main categories of CRC screening tests: endoscopic tests and stool-based tests. Visual tests are tests that use internal imaging to detect CRC, for example a colonoscopy. A colonoscopy is considered the gold standard of CRC diagnosis, a positive result in all other tests is verified by a colonoscopy. Nevertheless, it should be noted that colonoscopies have a very small, but not entirely insignificant, miss rate, which depends on the size of the adenoma (Van Rijn et al., 2006; Kaminski et al., 2010).

Stool-based tests are designed to detect certain chemicals in the patient's stool. Within this category, the Fecal Immunochemical Test (FIT) measures the presence of blood in stool. The FIT is found to be more effective at detecting CRC in the population than other Fecal-based tests.(Toes-Zoutendijk et al., 2017; Van Rossum et al., 2008; Hol et al., 2010). A FIT test may have a varying sensitivity and specificity. For example, an often-used cut-off value for the FIT is 10 μg Hg/g, denoted as FIT10. This test will give a

positive value if Faecal occult blood > 10 μg Hg/g and negative otherwise. However, the actual cut-off value may be varied by researchers, for example in cost-effectiveness (CE) studies (Ykema et al., 2020).

In a screening programme, at any time, an individual may be invited to none, one or multiple screening tests. For the general, low-risk population, intervals between tests are usually at least a year. We refer to the totality of assignments over an individual's lifetime as a 'screening programme'. Screening may lead to the detection of adenomas or CRC, which allows for the initiation of removal of localized lesions, in case of adenomas or early stage CRCs, or the earlier initiation of treatment, in case of non-localized CRCs.

Despite the merits of screening programmes, there are limiting factors that motivate policy makers to enforce restrictions on screening programmes. These include required effort, anxiety, complications and other negative externalities in the population and an economic burden on the parties that conduct the screening, which are often paid for by public funds. While the benefits outweigh the costs in those that actually develop CRC, 'unnecessary' screenings should be minimized.

2.2 MISCAN-Colon

A way to simulate the harms, costs and benefits of CRC screening is through microsimulation models. These have been shown to be reliable tools in determining optimal screening protocols and predicting future outcomes of CRC screening. The MISCAN (MIcrosimulation SCreening ANalysis)-Colon (Loeve et al., 1999) is one of those well-established stochastic microsimulation models, used to simulate the development of CRC and how it is affected by screening, in individuals in a population. Hence, it guides policy-makers worldwide (Cenin, St John, Ledger, Slevin, and Lansdorp-Vogelaar (2014) for Australia, Rutter et al. (2016) for the US, Van Hees et al. (2015) for the Netherlands and Goede et al. (2015) for Canada.
In their seminal paper, Loeve et al. (1999) describe in detail the internal functioning of MISCAN-Colon. We will summarize the dynamics here. For each individual, the model draws a random date of birth and death (assuming no CRC), using birth and life tables (representative of the population under consideration). As each simulated person ages, none, one or more adenomas may develop in these individuals, as seen in figure 1. These adenomas (progressive or non-progressive) can grow in size from small ($<$5 mm) to medium (6-9 mm) and then to large ($>$ 10 mm). In case of progressive adenomas, those may also progress into preclinical cancer and in stage from I to IV. Nevertheless, during each stage, CRC might be diagnosed because of symptoms and, after clinical diagnosis, a corresponding survival time is assigned. For synchronous CRCs, the survival is set on the most advanced cancer. The date of death for CRC patients is the earliest simulated date of death (due to CRC diagnosed versus another cause). Screening has the power to modify some of the simulated life histories. This is because some CRC may be prevented by the earlier detection/removal of adenomas.

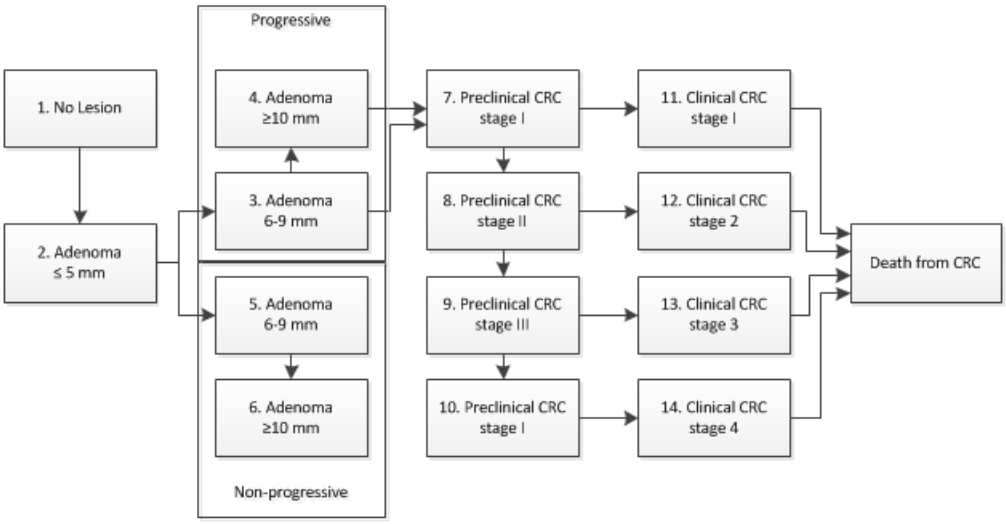


Figure 1: Progressive stages of CRC, as modeled in MISCAN-Colon

Individual Risk

The predisposition of an individual to develop an adenoma is governed by their individual hazard rate. The hazard rate is calculated from their age and a risk index that is drawn from a Gamma distribution, for each individual separately. The shape and scale parameters, k and θ, of the Gamma Distribution, which determine its mean and variance, are carefully calibrated based on adenoma prevalance and multiplicity and may vary per population. The risk scale is continuous and values can be seen as an odds ratio of the mean. The mean and variance of risk in the general US population are considered to be 1 and 1.9863 respectively (Knudsen et al., 2016), which corresponds to a $\Gamma(k = 0.503454, \theta = 1.98628)$ distribution. This is illustrated in figure 2. For metamodeling purposes, we want to set bounds on the risk variable. The shape of the Gamma distribution implies that high risk individuals are very rare in the population. In fact, approximately 70% of the population has individual risk lower than the mean of 1 and 98% is expected to be lower than 5.

Screening Intervention

In MISCAN-Colon, screening tests are user-defined interventions, with specific parameters per screening test. Because, in MISCAN-Colon, the time-component of a single screening intervention is a continuous variable, a screening protocol may theoretically be an infinite range of random time points in a simulated person's life, with assigned intervention. To make this more tractable, researchers may restrict this range by binning the age range, for example in year-bins. Even more generally, researchers often prefer to summarize a screening protocol by means of high-level variables, such as a starting age, a stopping age and a frequency of applying a certain test. For example, the current screening policy in the Netherlands is defined by a biennial FIT test, between the ages of 55 and 75. This,

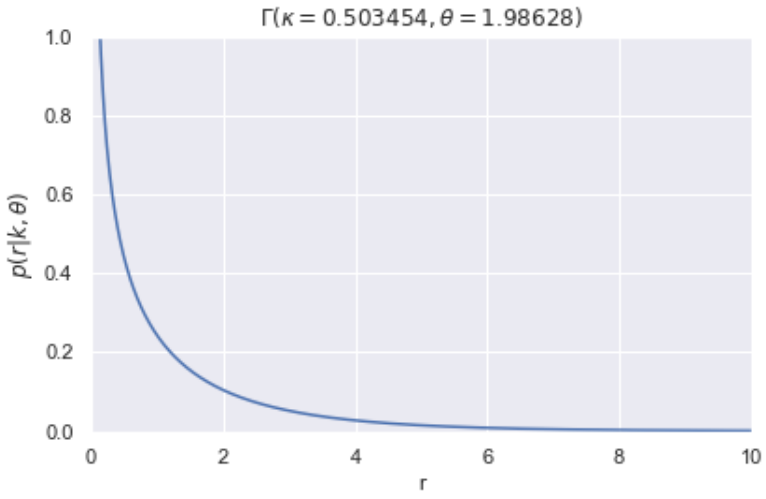


Figure 2: Probability density function of Risk in a US population

however, assumes that a constant testing frequency is optimal, which is counter-intuitive, as it is known that age influences the hazard rate of adenoma onset and life years to be gained from screening.

Stratification of the population

Because of the skewness of the Gamma distribution, with the majority of individuals occupying a small part of the risk range, CE modelers should be able to simulate population strata of variable size. To facilitate this, ideally, inputs and outcomes should be observable on an individual level. MISCAN-Colon allows for stratification of the population, where each stratum may be assigned separate risk parameters and screening programmes. While the input parameters allow for stratification, the output parameters are only presented for the entire population. While these are tracked by the model, the current, publicly available version of the model does not return them. In addition, the risk index per individual is not observable. This way, we cannot observe risk and outcomes on an individual level, which we require. In section 3.3, we present a workaround for these limitations.

The scenarios run through MISCAN-Colon are often compared based on two outcomes, Life Years Gained (LYG) and cost. Scenarios can be compared by plotting these outcomes for different strategies to form a cost-effectiveness curve.

2.3 Uncertainty dynamics of MISCAN-Colon

We require an understanding of MISCAN's dynamics to make decisions for our metamodel. For example, we want to get an idea of the variance in the output and how the mean and variance in outcomes are affected by the input variables. In the literature, there is no study that reports the input-output dynamics of MISCAN-Colon. Therefore, we have conducted a simulation study of MISCAN-Colon. The full results can be found in section Appendix

D, the most important findings are summarized here.

As can be expected from a microsimulation model, we observe that the mean and variance of observations scale linearly with the population size. This is an important finding, as we wish to sum observations of outcomes for population sub-samples later (Section 3.3). For risk, we have several important findings. First, we notice that the outcome of LYG and Cost increases when risk increases, if screening is held constant. While this is easily guessed, we also show that the increase is smooth and sub-linear. This can help us in determining the metamodel infrastructure. In addition, we notice that variability (measured by the coefficient of variation) decreases for higher risk individuals. This may have implications for the sampling distribution we choose (section 3.3). For screening, we notice a similar pattern, where more intense screening yields higher observations of LYG and Cost, but decreased variability in Cost. The latter may be explained through the fact that, given a certain proportion of sick people, there is less uncertainty in the disease being caught and therefore less variety in the costs, within the population of sick people. It does not as strongly influence the variability of LYG.

2.4 Metamodels

Now that we have established the dynamics and relevance of the main model, we will now discuss the requirements and characteristics of the metamodel that will emulate it.

Definition of a metamodel

In the sense that a model is by definition an abstraction of reality, a metamodel is an abstraction of a model, a model of a model. Metamodels are often employed when a high amount of model runs is required, for example for sensitivity analysis, but this requires too much computational resources (Villa-Vialaneix, Follador, Ratto, & Leip, 2012). A metamodel's aim is to emulate the main model's mapping from model input to model output, by training it on a limited amount of model runs. By only mapping this inferred relationship and not replicating the main model's complex calculations, this may lead to a shorter run time. A metamodel may be employed for four different goals, understanding of a main model (or underlying problem entity), prediction of new outcomes, optimization or Verification and Validation (V&V) (Kleijnen & Sargent, 2000).

In essence, any ML framework that maps some input to some outcome, can be used as a metamodel. Popular choices are Radial Basis Functions (RBF) (Sóbester, Forrester, Toal, Tresidder, & Tucker, 2014), Kriging (Freedman, 2009), Polynomial methods (Hussain, Barton, & Joshi, 2002), Support-Vector Machines (SVM) (Lai, Yu, Huang, & Wang, 2006), Random Forest (RF) (Breiman, 2001) and Artificial Neural Network (ANN)s. Appendix C provides an overview of pros and cons of the different models.

As can be expected, drawbacks of metamodels are analogous to the limitations of the main model, when compared to reality. The capabilities of the metamodel are limited by the constraints set by the modeler, both in terms of variable selection and selecting

the training data. Therefore, a correct model selection process and Design of Experiment (DoE) are vital steps towards building a fast and accurate metamodel.

Formally, we train the metamodel to predict outcomes for a certain scenario of interest. Let there be a data set of input data points, denoted by $x_i \in \mathbb{R}^k$ and let the main model calculate the associated outcomes, denoted by $f(x_i) = y_i \in \mathrm{R}^l$. The metamodel's aim is to generate estimations $M(x_i) = \hat{y}_i \in \mathrm{R}^l$, such that $\hat{y}_i$ is adequately similar to y_i. Adequacy of similarity is then represented by the Loss function (also known as Error function), $L(\hat{y}_i, y_i)$.

It is well-known that there are many different loss functions and that the decision for a loss function is based on the modeler's design requirements. One of the most common is the Mean Squared Error (MSE), which is calculated through

$$MSE(\mathbf{W}) = \frac{1}{n}\sum_{i=1}^{n}\|\hat{y}(x_i, \beta) - y_i\|^2,$$

also known as quadratic loss or L2 loss. An alternative exists in the form of the Mean Average Error (MAE), denoted by

$$MAE(\mathbf{W}) = \frac{1}{n}\sum_{i=1}^{n}\|\hat{y}(x_i, \beta) - y_i\|,$$

which is also referred to as L1 loss. The main difference between these is the fact that the MSE minimizes the squared error, instead of the absolute error. This implies that outliers are punished more heavily and have a larger effect on the correction of the weights. In case of uninformative outliers, this may have a negative effect on the model and the model will overestimate the MSE. MAE, on the other hand, is more robust to outliers. However, all loss is weighted equally and additive, regardless of the magnitude of the loss, which may cause the model to improve estimates in a certain range by underperforming in a different range. This compensatory behavior is not encouraged by the MSE.

2.5 Methods for Data Sampling

An important part of training a meta-model is generating the right data from the main simulation model. Because the main model is assumed to be expensive to run, choosing the right sampling method is necessary to let the model learn as efficiently as possible. A well-known way of covering a representative portion of the input space is an m^k (also known as full factorial) design. It aims to create representative input spaces by dividing the domain all k input variables into m levels, each unique scenario representing a data point. The main disadvantage of this method is the large number of scenarios to evaluate, as the data required increases exponentially with the amount of input variables. For MISCAN-Colon, this may range from 7 to several hundreds, depending on the problem definition by the modeler. As an alternative, oftentimes Latin Hypercube Sampling (LHS) (Morris & Mitchell, 1995) is employed. In LHS, all design variables are divided into partitions of a certain range and for each candidate solution, a value is sampled from the range.

This way, not all possible scenario is explored, but the solution space is still well-explored. Generally, this leads to lower correlation in the datapoints than when completely random sampling is used. However, Syberfeldt, Grimm, and Ng (2008) have shown that in some situations, full random sampling may lead to better results when training an ANN, than LHS.

To use this metamodel for population inference, it is required to estimate a weighted sum of the metamodel's inferences (this is further explained in section 3.3). Therefore, the possibility exists that the metamodel's optimal sampling distribution is not only achieved by a space-filling design, but the weighing of the observations should also be taken into account. For example, for a population in which the risk distribution is skewed strongly to the left, it might benefit the accurracy of the final estimate to have a metamodel that is more strongly tuned to lower risk. This, in turn, may be achieved through a sampling distribution that favors low risk. In other words, the training distribution that matches the distribution of the validation scenario, which adheres to standard practices in Machine Learning.

On the other hand, this causes a sparsity of observations in the high-risk range of the model. If we need a high amount of observations to accurately tune the model, this will cause the model to be inaccurate in the ranges that are sparsely populated by training observations. While the performance in the weighted sum results important nuances of idiosyncratic behavior will be lost.

Ideally, data is sampled uniformly, such that the model becomes accurate for the entire data range. However, when the size of the dataset is insufficient, it would be better to focus the sampling of the data on those risk ranges that will be weighed the heaviest in the final estimate for the population. These options will be further explored in 3.3.

2.6 Neural Network Framework

In this section, the theoretical background of Artificial Neural Networks will be provided, a motivation for this method and how it will be applied in this thesis is provided in section 3.3.

An ANN is a computational model, loosely inspired by biological neurons, that maps outputs Y to inputs X through a network of nodes and vertices. This mapping is performed through computational units on the nodes and vertices, Activations and Weights respectively. For simplicity's sake, we will explain the structure of a single-layer neural network here. The values at each hidden layer node z_j is calculated by performing an activation function on the activation value a_j calculated for that node. The activation value of the node is obtained by taking the propagated values of each input node x_i connecting to z_j with the weights of their connecting vertex, $w_{i,j}$, and then summing them. This can be formalized as

$$a_j = \sum_{i=1}^{k} w_{i,j} * X_i, \tag{1}$$

where a_j is the activation value at the node and X_i is the value of input node i. Consequently,

$$z_j = c(a_j), \tag{2}$$

where c is some activation function. Values at the output nodes are calculated in a similar fashion, only replacing x_i by z_j and z_j by y_k. Some value y_k can then me mapped to the input values through

$$\hat{y_k} = c(\sum_{i=1}^{k} w_{i,j} * X_i). \tag{3}$$

Using the same computational scheme, more hidden layers can be added, which allows more the mapping of more complex behavior. The mapping is then computed by

$$\hat{y_k}(X_i) = \sum_{j=1}^{k} w_{i,j}\ c(\sum_{s=1}^{p} w_{j,s}\ c(\ ...\ c(\sum_{l=1}^{o} w_{s,l} X_i)..)), \tag{4}$$

which can also be expressed more conveniently in matrix notation as

$$\hat{y_k}(\mathbf{x}) = c(\ ...c\ (c\ (\mathbf{x}\mathbf{W}_1)\mathbf{W}_2)...)\mathbf{W}_L, \tag{5}$$

where $\mathbf{x}$ is the $k \times 1$ vector of input variables and $\mathbf{W}_l$ the $p \times s$ weight matrix layer, mapping the node vector of layer $l-1$ with p nodes to the nodes of layer l with s nodes.

Activation function

The activation function c is used to scale propagated values at each node and help the ANN to model non-linearity. Without activation functions, an MLP is simply a linear regressor. In the last few years, the most popular activation function has been the rectifier (Ramachandran, Zoph, & Le, 2017), first used in deep learning by Glorot, Bordes, and Bengio (2011). A unit employing the rectifier is referred to as a rectified linear unit (ReLU). Mimicing the threshold activation behavior of certain biological neurons, it is also more biologically accurate, than other activations functions, such as the hyperbolic tangent and sigmoid neurons. Furthermore, it is computationally less expensive. Formally, the rectifier calculates the output of a neuron through:

$$z_j(a_j) = max(0, a_j).$$

In contrast to other activation functions, ReLU activation suffers less from saturation or the vanishing gradient problem (Goodfellow, Bengio, & Courville, 2016), both of which hamper learning by the network. A weakness of the ReLU is that, in some situations, a neuron may get stuck at a value of 0. Because in gradient optimization, its functional gradient also equals zero, it will no longer be updated. This is reffered to as a 'dying

ReLU', from which it is unlikely to recover from this (Lu, Shin, Su, & Karniadakis, 2019). This issue is mitigated through a smaller learning rate or asymmetric weight initialization.

Network Training and Optimization

The goal of training a Neural Network is to minimize the difference between estimates and actual values. This difference is calculated through the error function $L(\mathbf{W})$ (also known as the loss function).

The loss function is minimized by adapting the estimates to the true values, which is done by updating the weights matrices $\mathbf{W}$, in the model. We can formulate optimizing the model as the minimization problem

$$min_{\mathbf{W}} \nabla L(\mathbf{W}),$$

where ∇ denotes the gradient. This can be a challenging task, with a multitude of practical issues (including the aforementioned dying gradient or the exploding gradient problems). In Machine Learning literature, many different minimization algorithms are known, which vary in how they use the gradient and in the use of batches of observations for updating weights. An important variable is the learning rate λ, which determines the magnitude in which weights are updated based on observations. The lower the learning rate, the slower, but the more stable the model learns. One of the most popular and consistently used optimization algorithms is Adam (Kingma & Ba, 2014), which combines favorable properties of RMSprop (Tieleman & Hinton, 2012) and Stochastic Gradient Descent (SGD) (Robbins & Monro, 1951), it also incorporates advantages of Adagrad (Duchi, Hazan, & Singer, 2011). An in-depth explanation of Adam or either of its predecessors is beyond the scope of this paper, but its properties are interesting. Adam is an adaptive learning algorithm and finds individual learning rates for all parameters individually. Furthermore, it uses a moving average scheme that incorporates recent gradient information to update weights, which makes it more resistant to noisy data. To use Adam, setting four hyperparameters, parameters that are inherent to the model and not tuned to the data, is required. Kingma and Ba (2014) recommend $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-8}$, which generally perform well on a wide range of problems.

Model complexity and Regularization

In this section, we discuss modeling choices that govern complexity and regularization of Neural Networks. It is a well-known fact in statistical optimization that not one algorithm is superior in all aspects, for all data. This is often referred to as the No-Free-Lunch theorem (albeit a very loose interpretation of the seminal paper by Wolpert, Macready, et al. (1995) and Wolpert, Macready, et al. (1997). Adding to this, with its complex structure, ANNs are notoriously hard to interpret without meta-analysis. Because the correct structure of a Neural Network and the correct tuning of its hyperparameters are crucial precursors for its effectiveness (Goodfellow et al., 2016), evaluating the right modeling choices may make or break the effectiveness of our model. In this section, we evaluate

different modeling choices for Neural Networks and explain how we apply them.

Neural Networks are very well-geared towards learning complex, non-linear relationships between inputs and outputs. The more layers and the more nodes per layer an ANN has, the more complex learned relationships might be. The number of hidden units is an important decision in the bias-variance trade-off in Machine Learning. Typically, when the number of hidden units increases, the bias decreases and the variance increases in the model (Geman, Bienenstock, & Doursat, 1992). However, as in all forms of learning, too much flexibility in the learning structure, makes a model vulnerable to overfitting: the model learns characteristics of the data that are idiosyncratic to the training set, but not representative of the data generating process. This leads to poor generalization performance (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Deeper nets are especially vulnerable to this. Caruana, Lawrence, and Giles (2001) argue that overfitting in Neural Networks trained through gradient methods, such as Backpropagation, is not dependent on size, but agree that overfitting is an issue. Although many methods have been developed to prevent overfitting, one would ideally average weights over multiple networks, but this requires training and tuning many networks. Dropout offers a cheap approximation of that mechanism by probalistically removing nodes from the network for one training iteration. Therefore, the value at node i in layer l is denoted by

$$a_i = \sum_{i=1}^{k} w_{i,j} * x_i * r_j^l, \tag{6}$$

where $r_j^l \sim Bernoulli(p)$ and p is the dropout probability for layer l. This methods offers both model averaging and model regularization (Baldi & Sadowski, 2013).
Another issue with training deep Neural Networks concerns the simultaneous updating of layers. During training, Backpropagation calculates a gradient for each layer individually, under the assumption that all other layers maintain the same distribution. However, since all layers are updated simultaneously, this assumption no longer holds (Goodfellow et al., 2016). Since the distribution of inputs of a certain layer change as the weights of the previous layer change, the calculated gradient values are no longer 'up-to-date', which is referred to as the Internal Covariance Shift (Ioffe & Szegedy, 2015). Ioffe and Szegedy (2015) therefore propose Batch Normalization (BN). BN aims to prevent the Covariance Shift by standardizing the activations of the previous layer by mini-batch. Because the assumptions of Backpropagation now hold (approximately), this may increase training efficiency (Goodfellow et al., 2016). The effectiveness of BN has been showcased in research (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016; He, Zhang, Ren, & Sun, 2016; Amodei et al., 2016).

For the amount of nodes per hidden layer, it is common practice to only consider different powers of 2 as architectures. In unoffical sources, it has been suggested that this achieves best performance, but the literature provides no hard proof. The optimal amount of nodes per layer is not known a priori and can only be determined by experimentation, often informed by intuition or similar problems (Goodfellow et al., 2016).

2.7 Quantifying Uncertainty in Neural Networks

In (meta)modelling, it is often necessary to provide estimates with a measure of certainty, such as a prediction interval. Especially when the underlying data or model are stochastic. This section provides an overview of different types of uncertainty and how they may be estimated by Neural Networks.

Different types of Uncertainty

In a prediction interval, it is not clear whether uncertainty arises from inherent noise in the training data, or whether the uncertainty arises from modelling assumptions or the samplesize. Towards this purpose, we may distinguish epistemic and aleatoric uncertainty (Fox & Ülkümen, 2011). Aleatoric uncertainty, or statistical uncertainty, is inherent to the data generating process and is simply a quantification of the variability observed when repeating analysis of the same scenario. Epistemic uncertainty, or systematic uncertainty, is all uncertainty that is introduced in the measuring or modelling process and can be attributed to modelling assumptions or data selection. Theoretically, epistemic uncertainty can be reduced to zero with perfect knowledge of the system, while aleatoric uncertainty cannot. Generally, it is the modeler's goal to minimize the epistemic uncertainty, such that the total uncertainty approaches the aleatoric uncertainty. The modelling of epistemic uncertainty can be very useful in the metamodelling process, as it gives us an indication of the possible room for improvement of the model configuration or training data.

One way to model uncertainty in ANN outputs, is by replacing the scalar values of weights in the Neural Network by prior distributions and updating the parameters of these distributions by training. This is an example of Bayesian Learning and an ANN trained this way is referred to as a Bayesian Neural Network (BNN) (Neal, 2012). In short, BNNs try to learn the posterior distribution of the weights of the distribution parameters of the weights conditional on the data, $P(\theta|X, Y)$.

In recent years, many frameworks have been proposed to efficiently estimate BNNs, some of the more notable are Neal (2012); Kingma, Salimans, and Welling (2015); Gal and Ghahramani (2016); Lakshminarayanan, Pritzel, and Blundell (2017). And new frameworks keep being developed. Due to its simplicity and generalizing capabilities, Monte-Carlo Dropout (Gal & Ghahramani, 2016) has been receiving positive attention and has proven to deliver strong practical results (Zhu & Laptev, 2017).

For the interested reader, an intuitive explanation of how Monte-Carlo Dropout may be used to estimate a predictive distribution, is presented in Appendix F. However, this knowledge is not required to understand the methods used in this thesis.

Despite the scheme's simplicity, it does require the tuning of two additional hyperparameters, τ, the model precision, and p, the dropout probability. These parameters are tuned such that the log-likelihood between the predicted outcomes and observed outcomes is maximized. Because this log-likelihood is not necessarily convex for these parameters, we may use Bayesian Optimization or grid search to find optimal values. The optimization

curve is expected to be multimodal (Gal & Ghahramani, 2016).

We can further distinguish homoscedastic aleatoric uncertainty, which is general for the model, and heteroscedastic aleatoric uncertainty, which varies per observation, depending on the independent variables. Heteroscedastic uncertainty can be modeled by making the precision τ a function of the data during the calibration process, simply replacing it by some funcion $g^w(X)$ (Gal & Ghahramani, 2016), where the w indicates that it is dependent on the model weights.

3 Methods

The methods section will lay out the experimental framework of this thesis. This work will loosely follow the methodology by Kleijnen and Sargent (2000) and Sargent (2010) for the design and validation of simulation (meta)models. Based on these works, the methodology section consists of four subsections. First is the Purpose and Requirements section, which contains the intended use and design requirements of the metamodel, section 3.1. Then, the input and output variables are defined, in section 3.2. Then follows the Design of Experiment, which functions as a schematic of the metamodel, from data to results, section 3.3. Finally, an approach is required to validate the metamodel's effectiveness, which is presented in section 3.4.

3.1 Purpose and Requirements

In this section, the metamodel's primary goal is discussed. In their work, Kleijnen and Sargent (2000) distinguish between four different general goals for metamodels: Understanding, Prediction, Optimization and Validation & Verification. In the following paragraphs, we will explain how our metamodel will be used for prediction.

In CRC Screening literature, researchers may design an experiment in which different combinations of experimental variable settings, such as screening frequency and population risk, are simulated. Because simulations are expensive, only a low amount of different scenarios may be explored, before simulation time becomes intractable. With several experimental variables and those variables having several levels, this number may easily be reached before the researcher's curiosity is satisfied. The goal of the metamodel is to increase the number of scenarios that can be explored in the available time, by decreasing the time required for a single run. Inherently, the metamodel is an abstraction of the main model and is therefore an approximation.

As designers of the metamodel, we envision its role as supplementary to the main model, MISCAN-Colon. As an approximation of the main model, it will inherently have equal or more uncertainty in its estimates. This additional uncertainty is referred to as epistemic uncertainty, uncertainty attributable to the modelling process. Therefore, we advise that the metamodel be used for the quick exploration of a large range of scenarios. The best candidate results from that analysis can then be verified with the main model. As such, it aids in the more effective use of MISCAN-Colon.

Design Requirements

From the purpose of the metamodel above, follow five requirements: Speed, Accuracy, Uncertainty, Stratification and Balance. Now will follow in short how these requirements are formulated and when they are met.

To make up for a loss in accuracy and time required for training, the metamodel should be several times faster than the main model. While it is not feasible to evaluate every possible scenario, which would require 3^{61} evaluations (assuming three screening levels per year and 61 screening years), it could give an effective overview to at least make a prediction of all unique screening programmes obtained by all possible combinations of start ages, stop ages and frequencies for FIT and colonoscopy, e.g. those specified in section 3.3.1. This yields approximately 150,000 evaluations. Based on empirical tests, we estimate that this would take MISCAN-Colon 520 days, at 5 minutes per run. To make this tractable and to allow for sensitivity analysis, this should be reduced to days. If the metamodel is able to make a prediction per second, the run time would be reduced to 1.7 days. With no benchmarks available in the literature, we believe this to be a reasonable improvement.

Since MISCAN-Colon itself is stochastic, it makes sense to define accurracy in terms of statistical deviation, rather than a percentage value. Therefore the metamodel's aim is to make predictions that are within the Confidence Interval boundaries of the model. These confidence intervals are presented together with the results, in section 4.4.

We have established that stratifying the population based on individual risk to assign screening programmes more specifically can lead to gains in CE. Thus, the metamodel should be able to make model strata within the population, in which different strata undergo different screening protocols, depending on their risk.

An important requirement for the aforementioned stratification to work, is that the model's accuracy is balanced across the risk spectrum. A model that predicts well on low risk values, but poorly on high risk values, may still be considered accurate when making predictions on an entire population, if that population has mostly low-risk individuals. However, when a stratum has strictly high-risk individuals, the hypothetical model would poorly estimate the outcomes. Results for this requirement are presented in section 4.3.

Finally, because of its role in scientific research, the metamodel should not only return an estimate of the mean, but also of the uncertainty of its estimate. Preferably, the model should return both the aleatoric uncertainty and the epistemic uncertainty, which may be used to construct confidence and prediction intervals. If the metamodel is able to return a confidence interval of MISCAN-Colon, this would replace many runs of the main model, thereby multiplying the metamodel's relative efficiency.

In summary, the model should be fast, accurate, permitting stratification, balanced and return uncertainty. After presentations of the results, these requirements will be revisited in section 5.2.

3.2 Input and Output Variables

Now that we have established the core functioning of the metamodel, this section will discuss the different variables. In this thesis, every datapoint represents the input and output variables of a small, homogeneous population. This will further be explained in section 3.3. From hereon, we therefore consider the input variables per sub-population i.

Input Variables

We define the set input variables of the metamodel as X. X consists of a Risk component R and a Screening component S, as described in section 2.2, so $X = \{R, S\}$. For each sub-population i, a single data point may be generated, with values $X_i = \{r_i, S_i\}$.

Domain of individual Risk

The individual risk number is theoretically unbounded if the risk distribution from which it is drawn is unbounded, which is the case in the often-used Gamma distribution. However, in its role as an odds ratio, this is practically unlikely. We define the risk variable as

$$\{r_i \in \mathbb{R} \mid 0 < r_i \leq 25\},$$

where r_i is the individual risk of some sub-population i. This means that, while there is still a non-zero probability of values occurring outside this range, the metamodel will be trained and tested within it. The metamodel assumes the risk values of the population are already drawn.

Domain of Screening

We determine the cost-effective screening range to be between 25 and 85 and therefore assume that screening outside of this range may be clinically effective, but not interesting cost-effective for cost-effectiveness studies. A screening age of 25 is unprecedentedly low, as most studies do not start scenarios until age 50, but recently it has been found that CRC incidence is increasing in lower ages (<40) (Siegel et al., 2017). An upper bound of 85 is used by most recent CE studies (e.g. (Knudsen et al., 2016; Peterse et al., 2018)). In these studies, cost-effective strategies have stop ages lower than 85, but we cannot exclude the possibility that such a cost-effective strategy exists, as the studies are limited in their capacity to evaluate unique screening scenarios (which emphasizes the need for a metamodel). Assuming year-bins, this constitutes 61 screening variables. It should be noted that is considered too high-dimensional for some sampling methods, without dimension reduction.

We bin the time range of the screening protocol into bins of one year. Each year is then represented by one categorical variable, which denotes the screening test conducted. Based on 2.2, we define the range of screening variables as

$$S_i = \{s_1, s_2, ..., s_T\}, \forall t, t \in [1, ..., 60], s_t \in \{\text{None, FIT, Colonoscopy}\} \ ,$$

where S_i is the screening protocol assigned to some sub-population i. To let our metamodel handle these categorical variables, we employ one-hot encoding, which transforms each categorical variable into a number of binary variables, one for each level. With three levels per screening variable, this results in a total of 180 screening variables.

Output Variables

We define the vector of output variables of the metamodel as Y. Y consists of a LYG component L and a Cost component C, so $Y = \{L, C\}$. For each sub-population i, a single output may be predicted, with individual values $Y_i = \{L_i, C_i\}$.

LYG

Life-years gained (LYG) consist of a single random value for each observation, directly returned from the main model:

$$L_i \in \mathbb{R}$$

Cost

The Cost value is a vector containing different measures of cost outcome from the model. MISCAN-Colon returns values for random variables that have an associated cost. During post-processing, these values are multiplied with their variable cost to obtain the total cost per cost variable in the model. We consider those post-processed cost values to be the output of the model:

$$c_i \in \mathbb{R}^{29}$$

and $c_{i,j}$ is the cost associated with cost measure j for sub-population i. In this thesis, cost is measured in US Dollar, as variable cost values were adopted from Knudsen et al. (2016).

3.3 Design of Experiment

This section offers an overview of the steps required to sample data and obtain the metamodel. The following paragraphs provide an overview of the design, after which more detailed steps follow.

As mentioned before, with MISCAN-Colon as a black box, we cannot observe risk on the value of the individual. However, we can control the how the model draws risk by running MISCAN with a pre-specified mean and negligibly low variance (e.g. 1×10^{-20}). Then, we can simulate outcomes for small, homogeneous populations, i.e. the individuals share the same risk number. By drawing the input means of these homogeneous sub-populations from the population risk distribution we want to model, we can add the outcomes together and as such infer a discretized estimate for the entire population. Appendix B provides a proof by simulation for this scheme.

As such, the metamodeling process consists of two steps: first, estimation of the outcomes for homogeneous sub-populations of a population and second, combining those outcomes into an estimate for a complete, heterogeneous population, i.e. the discretization process. These are further divided into four experiments. The upcoming section will discuss the series of experiments that are required for these four steps. First, data is generated, which is described in section 3.3.1. Then, the first experiment follows in section 3.3.2, here is described how to find an optimal Machine Learning Framework for ANN estimates. Experiment 2 aims to find optimal hyperparameters through Bayesian Optimization, which is found in section 3.3.3. Section 3.3.4 describes how to make predictions using the BNN, which is experiment 3, and section 3.3.5 presents the discretization algorithm required for the final metamodel's predictions, which is experiment 4. Finally, the last section describes how all experiments are evaluated, section 3.4.

3.3.1 Data

The metamodeling process requires training data and test data. Because this thesis compares different sampling distributions for metamodel development, a training data set will be uniquely generated for each individual sampling strategy. The test data will function as a ground truth to validate the metamodel's estimates by and will be generated by running MISCAN-Colon (the main model) a high number of times. In contrast to the training data, the test data can be considered a universal truth and will be the same for validation of the models from all sampling strategies. The test data will consist of two separate datasets, one for validating the BNN estimates on small, homogeneous sub-populations; and one for validating the estimates on entire populations.

Data was generated through MISCAN-Colon. Calibration settings were used for a US population. For the screening tests, the same settings for screening costs, specificity and sensitivity were used as in Knudsen et al. (2016).

Training Data

In section 2.5, we concluded that we would prefer uniform sampling in case of sufficient data, otherwise we would prefer sampling from a distribution that is more likely to emphasize realistic scenarios. For the risk variable, we compare a $Unif(0, 25)$ distribution vs. a $\Gamma(k = 0.503454, \theta = 1.98628)$ distribution, which represent balanced sampling and realistic sampling, respectively, as explained in section 2.5. In the same fashion, for the screening variable, a a fully random and a frequency-based scheme are compared. The random screening scheme is generated by randomly drawing a test for each year a screening could possibly be conducted. The frequency-based screening scheme considers all possible combinations of starting ages ($S_A \in \{25, .., 85\}$), stopping ages ($S_O \in \{S_A, .., 85\}$) and screening frequencies ($f \in \{0, .., 10\}$), for all tests ($T \in$ {FIT,Colonoscopy}). In total, this yields four different sampling scenarios, with two options for both Screening and Risk. To function as a benchmark, we add a more general method, orthogonal Latin Hypercube Sampling (Morris & Mitchell, 1995), which seeks to minimize correlation between the sam-

Strategy	Risk variable	Screening variables
1	Uniform	Random
2	Gamma	Random
3	Uniform	Frequency-based
4	Gamma	Frequency-based
5	Orthogonal	Orthogonal

Table 1: Different sampling strategies to generate data from MISCAN-Colon.

Label	Purpose	Representation	Level	Source
A	Training Data	Sub-population	Unique per sampling strategy	MISCAN
B-P	Predictions	Sub-population	Unique per sampling strategy	BNN
B-T	Ground truth	Sub-population	Universal	MISCAN
C-P	Predictions	Population	Unique per sampling strategy	Discretization
C-T	Ground truth	Population	Universal	MISCAN

Table 2: Overview of different types of datasets used. Data obtained from prediction has a label suffxed by '-P', while test data is suffixed by '-T'.

pled observations. An overview of the different sampling strategies is presented in table 1. The resulting training dataset for each sampling strategy is labeled as 'A', see table 2.

Test Data

Now, we discuss the test data sets. As was explained in the previous section, it is required to first make predictions on sub-populations. To validate these estimates, the true mean and variance for a such homogeneous sub-population are required, for all risk values of interest used for estimation. This is done through the same homogeneous sub-populations as used for the training data, only we now run MISCAN-Colon, for each risk value of interest, for both one low-intensity and one high-intensity screening scenario, 1,000 times. These thousand runs can be summarized in a sample mean and variance and act as a Monte-Carlo estimate for that specific risk value, for that screening scenario, i.e. a ground truth. This results in the first test dataset, B-T.
In the end, the purpose of the metamodel is to make predictions for complete, heterogeneous populations. Those predictions can only be validated by a ground truth data for these heterogeneous populations. To obtain this data, MISCAN-Colon is run to simulate a large, heterogeneous population, in which 10M people are simulated and their risk is drawn from $\Gamma(k = 0.503454,\ \theta = 1.98628)$ distribution. Again, the sample mean and variance are calculated and serve as a ground truth to validate the metamodel. This results in the second test set, C-T. An overview of the different datasets obtained through MISCAN is presented in table 2.

Now that the datasets are established, we explain how a metamodel will be trained for each of the sampling strategies in table 1.

3.3.2 Selection of metamodel framework

We have established that the main problem is high-dimensional and that we require estimates of the aleatoric and epistemic uncertainty. Therefore, we elect to use an ANN with a Monte-Carlo Dropout extension to make estimates of the mean and variance of the population, an extensive evaluation of the advantages and disadvantages of different NNs can be found in Appendix C.

The Bayesian Neural Network not only returns predictions of the mean of the outcomes of specific scenarios, but also of the associated epistemic and aleatoric uncertainty (see section 2.7). By modeling the epistemic uncertainty, we can test whether the sampling size and sampling scheme are sufficient for specific scenarios. This not only helps us to validate the metamodel, but also to help plan future steps. Modelling aleatoric uncertainty helps us understand the inherent variance in the sampled dataset. Together, they allow us to estimate the predictive distribution, which we can use to make confidence intervals for our estimates. One of the merits of Monte-Carlo Dropout BNNs is that they can be used as a simple extension of existing 'vanilla' Neural Networks. However, this means that we first need to find an optimal non-Bayesian Neural Network architecture per sampling strategy, before it can be optimized for uncertainty prediction.

It is common knowledge that, a priori, no de facto ideal approximation method can be determined without testing. Rather, the choice for a meta-model framework should be based on the model characteristics and shape of the data. For this thesis, this means that the model should be well-equipped to handle high-dimensionality and both continuous and categorical variables. Moreover, it should be able to make very accurate inferences on a wide range of scenarios.

Experiment 1: ANN Hyperparameter Optimization

We conduct a grid search considering all ANN design variables discussed in section 2.6, they are summarized in table 3. A grid search is considered an effective way of optimizing ANN architectures, as the search space can be considered to be discrete. This yields a total of 168 unique NN designs, which we train and validate for all sampling methods. To make all four experiments tractable, the models are trained and optimized on a maximum of 100,000 observations. Because the actual datasets are much larger, we validate the ANNs by making predictions on five test sets, that are independently sampled from the larger datasets.

To make training time and testing time tractable, we train on a random sub-sample (without replacement) of 100,000 observations. We validate each trained ANN on five different test sets of size 100,000 and calculate the MSE and SMAPE for each test set.

We then average over those five holdout sets, to get an average MSE and SMAPE score per trained ANN. While the datasets are larger, we chose to only sample 100,000 observations, otherwise later experiments become intractable. The process above is repeated for each sampling method.

Variable	Levels	
Nodes per Layer	$\{128, 256, 512\}$	
Hidden layers	$\{1, 2, 3, 4\}$	
Batch Normalization	{Yes, No}	if no, Dropout(0.5) layers are applied
Y	{LYG, Cost, Both}	
Cost Split	{No Split, Categories, Full Split}	if $Y \in$ {Cost, Both}
Cost Split	{No Split}	if $Y \in$ {LYG}

Table 3: Caption

Per sampling method, we select the optimal networks for estimating outcomes LYG and Cost, by selecting the network with the lowest MSE. These network designs will be used for the BNN module of the metamodel.

3.3.3 Experiment 2: BNN Hyperparameter Optimization

To correctly use an ANN with Monte-Carlo Dropout as a Baysian approximation, we require the tuning of three additional hyperparameters, the length scale l, the model precision τ and the dropout probability p. In contrast to the vanilla ANN hyperparameters, these are optimized over a continuous scale. As τ is a predictor of the model's aleatoric variance, it is important to find this value with high precision, which is not tractable through a grid search. Furthermore, we expect the predictive distribution of the BNN to be multi-modal (Gal & Ghahramani, 2016) and function evaluations are expensive, therefore we elect to use Bayesian Optimization (BO) to find optimal parameter values, as it employs a scheme that varies exploration and exploitation. For each BO iteration, we train a neural network with certain parameter values and train a Monte-Carlo Dropout BNN. With this BNN, we make MC predictions using the independent variables of the test set. We then calculate the predictive mean and variance and consequently the log-likelihood of these values compared to the dependent variables of the test set. The Bayesian Optimization algorithm iteratively maximizes this log-likelihood by varying the tuneable hyperparameters. For each sampling method, we run the BO algorithm until convergence, using the optimal ANN architecture found in the previous step. Bayesian Optimization algorithms have no formal convergence criteria (Scotto Di Perrotolo, 2018), so we run the algorithm a high number of times (300+) and conclude convergence when the log-likelihood top 5 observations are no more than 1 % apart. Runs continuing beyond these criteria have shown little marginal improvement. We use Tree-structured Parzen Estimator (TPE) and uniform sampling of the specified ranges. The ranges are set as $[0.01, 0.99]$ for dropout, which are theoretical boundaries, $[0, 100]$ for precision and $\{1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}\}$ for the length scale. To run the BO algorithm, the Hyperopt package (Bergstra, Yamins, & Cox, n.d.) in Python is used.

3.3.4 Experiment 3: BNN Prediction

To make the estimates more robust against selection bias and weight initialization, we combine the estimates of l iterations of fully training the BNN. Because the outcomes of

the BNN are drawn from a Gaussian distribution, we can consider the predictions estimates as a Gaussian mixture model (Cobb et al., 2019). We then calculate the ensemble mean as such:

$$\mu_{ens}(Y) = \frac{1}{L}\sum_{l}^{L}\mu(Y)_l \ \forall Y \in L, C,$$

in which $\mu(Y)_l$ is the estimated mean of the l^{th} BNN for outcome Y. Similarly, we calculate the variance of the mixture by the law of total variance (Cobb et al., 2019)

$$\sigma^2_{ens}(Y) = \frac{1}{L}\sum_{l}^{L}(\mu(Y)_l - \mu_{ens}(Y))^2 + \frac{1}{l}\sum_{l}^{L}\sigma^2(Y)_l,$$

in which $\mu(Y)_l$ is the estimated variance of the l^{th} BNN for outcome Y.

3.3.5 Experiment 4: Discretization

For the final experiment, we use a discretization scheme. First, we determine the risk distribution of CRC incidence of the population we wish to model. Then we draw a large number N random numbers from this distribution. We divide the random numbers into a M bins, such that $M << N$. The bins are based on quantiles of the risk distribution, so that each bin contains approximately the same amount of random numbers. Intuitively, this means that the bin around the mode will be the narrowest, such that the more populated areas of the curve will have more bins. For each bin, the average risk value is calculated, which results in the $M \times 1$-dimensional array $k = [r_1, r_2, ..., r_m, .., r_M]$. Each risk element r_m will be assigned a screening protocol S_m, together they will form the BNN's input scenario X_m. Each scenario X_m will be evaluated by the BNN, yielding a mean and variance prediction per outcome, for each risk element. Because we have shown in Appendix D that observations of MISCAN-Colon are independent of each other (as is also inherent to its micro-simulation nature), we can simply add up the sub-sample means and variances to approximate the population mean and variance. Hence:

$$\mu(Y) = \frac{1}{M}\sum_{m}^{M}\mu(Y_m) \ \forall Y \ \in L, C$$

and

$$Var(Y) = \frac{1}{M}\sum_{m}^{M}Var(Y_m) \ \forall Y \ \in L, C,$$

which represent the final estimates of the metamodel.

3.4 Validation

After establishing the variables of the metamodel, we now discuss validation criteria for the metamodel.

With validation of the metamodel, we run into two limitations. First, the data we use

to train the model cannot be used for validation, because each observation represents a unique and random sub-population scenario, not a heterogeneous population. Second, to generate validation values for both the mean and the variance of the model, we need to run MISCAN a high amount of times for every unique scenario, which is intractable. We cannot obtain the mean and variance of the main model through analytical means, therefore we conduct a Monte-Carlo simulation. We do this by running the same scenario through MISCAN-Colon a high number of times and recording the outcomes, from which we obtain the sample mean and variance. We assume these approach the true mean and variance for the outcomes of these scenarios, for large enough amount of observations. To minimize the correlation in observations, we initiate a new random seed for each iteration.

To test the validity of the metamodel, we try two test scenarios. In section 2.3, we conclude that the intensity of screening is an important factor for the scale and variability of outcomes of the model. Therefore, to explore both ends of the spectrum, we create two ground truths: one for a low-intensity screening and one for a high-intensity screening scenario. The low-intensity screening scenario will feature a 1970 US population with a bi-yearly FIT between 55 and 75, with a colonoscopy at 69. The high intensity screening scenario will be randomly drawn, with no test, a FIT10 test or a colonoscopy randomly drawn at each lifeyear in the screening range, 25 to 85. These scenarios will be run through MISCAN-Colon 1,000 times each, each time with a different random seed. We record the outcomes and calculate a sample mean and sample variance over them, which we will use as the ground truth outcomes.

Because all data is generated from statistical distributions, without external distortion, we assume the simulation data inherently contains no uninformative outliers. Furthermore, we want the metamodel to perform well across the entire spectrum of inference, therefore MSE seems like the better choice. Finally, it should be noted that optimizing on percentage errors may cause a bias in estimations (Kolassa & Martin, 2011).

Choice of accuracy measure

In section 2.4, merits of different error measures were discussed. We want the model to perform equally well across the outcome domain, therefore we use the MSPE to evaluate model accuracy. To serve intuition in model evaluation, we will also the SMAPE value.

Generally, for a simulation model to be valid, it needs to be accurate enough to correctly distinguish between adjacent scenarios, i.e. those scenarios that have similar outcomes when run through the main model. As such, the metamodel's prediction interval should ideally be narrow enough that it does not cover two adjacent scenarios. However, after thorough analysis of MISCAN-Colon we have learnt that some, non-identical scenarios may have at least one identical outcome, with a precision of at least 8 decimals. Therefore, in its supplementary role to the main model, which was described in section 3.1, it may be enough to be fairly accurate. For lack of benchmark in the literature, we

Experiment	Input Data	Output Data	Test Data	Description
1	A			ANN hyperparameter search
2	A			BNN hyperparameter search
3	A	BE	BT	BNN Inference
4	BE	CE	CT	Discretization

Table 4: Summary of experiments

will describe fairly accurate as the mean not being statistically different from the mean of the MISCAN-Colon estimate. Through a Jarque-Bera test, a non-significant p-value led us to conclude that we could not reject the null-hypothesis that the results from the Monte-Carlo simulation follow a Gaussian distribution. Because the metamodel's output is a summation of Gaussian processes, we can assume the metamodel's output also follows a Gaussian distribution. Therefore, we may use a two-sided t-test for the mean estimate of the metamodel and an F-test for the variance estimate.

To summarize, we will conduct two main experiments. The first one entails finding an optimal ANN architecture through grid search, which we will denote experiment 1. Finding optimal BNN hyperparameter values, we will refer to as experiment 2. Using the BNNs to make estimation is denoted experiment 3. Finally, aggregating the BNN values of experiment 3 into population estimates is experiment 5.

4 Results

In this section, the most important results will be discussed. The first part, will focus on the first experiment, the selection of the NN architecture, section 4.1. Next, are the results of experiment 2, the optimization of BNN parameters, section 4.2. After that, the results of experiment 3, the accuracy of the Bayesian Neural Networks, i.e. the accuracy of estimation of homogeneous sub-populations, in section 4.3. The fourth part and final experiment, 4.4, will focus on the accuracy of the entire metamodel, which combines the uncertainty of the BNN inference and that of the discretization process. For readability, we will present the most relevant plots and results here, the full results will be available in the Appendix.

4.1 Experiment 1: ANN Model Selection

In this section, we describe the results of training all Neural Network configurations on a random sample of 100,000 observations, for each data source, as described in section 3.3, i.e. experiment 1. The main question to be answered in this section is which ANN architecture is optimal for each sampling strategy. We have only included models that apply the natural logarithm to results, as models without it were consistently slower to train and took problematically long to converge. The best architectures per sampling strategy are summarized in table 5. In the appendix, the top 10 best performers for LYG and Cost, per sampling strategy are provided, in tables 17 and 18.

For both outcomes, MSE values of the top 10 smoothly increase, this could be an indication that the top performing networks are fairly interchangeable. This implies that a modeler may choose some other well-performing network because of favorable properties. For example, when using sampling strategy 1, a 4-layer network may be exchanged for a 1-layer network, at the cost of a small increase in MSE. This may be favorable, as shallow networks are often a smaller burden on computational resources than deeper ones. Other than that, across sampling strategies and within sampling strategies, network depth or the number of nodes per layer seems no definite determinant of ANN success.

Generally, the MSE scores increase smoothly when ordered, i.e. no sharp increases or decreases. This indicates that the NN architectures are fairly interchangeable and the results will not be too harshly impacted by changing the NN architecture.

NN architecture may be trivial across sampling methods, as their training data is sampled from the same main model. Since the sampling methods are independent, we use a Wilcoxon Rank Sum Test between the MSE values of different sampling methods. We do this for both outcomes, LYG and Cost. Per outcome, this entails 10 comparisons, so we apply a Bonferroni correction and set $\alpha = 0.005$). Only for outcome Cost, the Rank Sum test statistic is insignificant ($p = 0.34$), which cannot reject the null hypothesis that they are sampled from the same distribution. The opposite applies for all other comparisons.

LYG: Risk, Screen	id	layers	cost split	BN	Nodes	Both	MSE	SMAPE
1: Unif, Unif	165	4	no split	True	128	No	8.53×10^3	0.00079
2: Gam, Unif	145	1	no split	True	256	No	1.02×10^3	0.0063
3: Unif, Freq	166	4	no split	True	256	No	7.30×10^3	0.012
4: Gam, Freq	165	4	no split	True	128	No	7.94×10^2	0.030
5: Orth, Orth	144	1	no split	False	128	No	5.04×10^3	0.0032
Cost: Risk, Screen	id	layers	cost split	BN	Nodes	Both	MSE	SMAPE
1: Unif, Unif	85	1	no split	True	256	No	1.28×10^{12}	0.0010
2: Gam, Unif	87	2	no split	True	128	No	1.85×10^{11}	0.0039
3: Unif, Freq	88	2	no split	True	256	No	3.78×10^{12}	0.0098
4: Gam, Freq	93	4	no split	True	128	No	1.31×10^{12}	0.030
5: Orth, Orth	86	1	no split	True	512	No	9.22×10^{11}	0.0014

Table 5: Selected Networks for LYG and Cost for different sampling methods

This test indicates that we cannot generalize NN architecture optimality across sampling methods and we have to optimize for each sampling method individually. Finally, there is no architecture that is in the top 10 for all strategies, for either LYG or cost.

The final, optimal NN architectures for each sampling method are presented in table 5. It should be noted that SMAPE and MSE values for each sampling method can be used to compare architectures trained for that sampling method, because they are calculated based on the same dataset. As such, these measures cannot be compared between different sampling strategies, as they are not based on datasets with different ranges and variances. We will compare the sampling strategies by means of a universal test set in experiment 3, section 4.3. Furthermore, the results in this table give no definite indication of metamodel accuracy, as the final test data (in experiment 4) is of a different shape than the training data for this experiment.

4.2 Experiment 2: BNN hyperparameter optimization

In this section, the optimal NN architectures for each sampling strategy, found in the previous experiment, were used to find optimal hyperparameters for BNN architectures, through Bayesian Optimization. The optimal parameters are presented in table 6. This table presents the combination of values that yielded the highest log-likelihood for each strategy, for both LYG and Cost. An overview of the five best values per outcome, per sampling strategy can be found in the appendix. Most notably, we see the lowest dropout values for sampling strategy 2, for both LYG and Cost. This follows from the fact that Other than that, we notice that the top results per sampling strategy are not similar, which indicates that a single, global optimum could not be found.

4.3 Experiment 3: Accuracy of Bayesian Neural Network

This section will isolate the results of the first part of the metamodel, the Bayesian Neural Network, before discretization. This means that the BNN's predictions on universal test set

LYG: Strategy	λ	Dropout	τ
1	1×10^{-4}	0.32	1.78×10^{-4}
2	1×10^{-2}	0.079	2.43×10^{-3}
3	1×10^{-3}	0.22	2.93×10^{-4}
4	1×10^{-3}	0.18	1.59×10^{-3}
5	1×10^{-2}	0.25	2.25×10^{-4}
Cost: Strategy	λ	Dropout	τ
1	1×10^{-4}	0.23	1.77×10^{-12}
2	1×10^{-4}	0.16	6.42×10^{-10}
3	1×10^{-3}	0.35	6.08×10^{-11}
4	1×10^{-2}	0.23	3.97×10^{-11}
5	1×10^{-2}	0.42	4.89×10^{-11}

Table 6: Optimal Hyperparameters for Monte-Carlo Dropout obtained from Bayesian Optimization

B-T will be evaluated. All measures of accuracy in this part of the Results section, are calculated by comparing the predictions on homogeneous sub-populations to those generated by MISCAN, i.e. test set B-T. These are all predictions on homogeneous sub-populations, with a population size of 1,000. The results are presented in figures 3 and 4.

Sampling Strategy 1

This strategy can be considered to have high risk and high-intensity screening on average. We see that the epistemic uncertainty is the lowest around 12.5, which is the mean of the $Unif(0, 25)$ distribution that was used to generate risk for this strategy. The SMAPE for high-intensity LYG predictions also approaches zero at this risk value. Surprisingly, we do not see the same behavior for high-intensity Cost predictions. Generally, we do see that the BNN performs best in high-intensity screening scenario, as expected. Furthermore, it heavily underestimates LYG in the low-intensity screening scenario, which was expected, as the observations of LYG in the training data were on average higher than the values in the test set.

Sampling Strategy 2

This strategy can be considered to have low risk and high-intensity screening on average. The Gamma distribution used to generate risk for this sampling strategy, has 60 % of observations lower than one and only 1 % higher than five. Therefore, it makes sense that we see epistemic uncertainty increase for higher risk, as there are fewer observations to train the BNN. We also see that the SMAPE values for high intensity screening scenarios are low, rarely surpassing 0.05. The only exception is the SMAPE for high intensity screening LYG predictions, which is very high. Nevertheless, this is observed for all BNN estimates and can most probably be attributed to the very high variability of LYG for low risk, see section D.3. For low risk, we observe very little epistemic, but also ensemble uncertainty. This is an indication that all BNNs trained on the data 'agree' on those predictions, which

could be an indication of a good fit.

Sampling Strategy 3

This strategy can be considered to have high risk and low-intensity screening on average. It is therefore surprising that the BNN performs weakly on the low-intensity scenarios. For predictions of high intensity Cost, we see very large ensemble uncertainty. This means that the different BNN ensemble estimates were far apart. This could be an indication of an ill fit.

Sampling Strategy 4

This strategy can be considered to have low risk and low-intensity screening on average. We would therefore expect it to perform best on these domains. For Cost, we see a consistent overestimation of the outcome, as compared to the true values. Unsurprisingly, we see the epistemic uncertainty increase for higher risk, where data is more sparse. This was also seen in sampling strategy 2.

Sampling Strategy 5

This dataset is similar to the data generated by sampling strategy 1, only with a mean of 15. Unsurprisingly then, the results are almost identical to those of sampling strategy 1. It should be noted that we see the epistemic uncertainty almost disappear around 7.5, which is an indication that both parameter and data uncertainty are almost zero, which is an indication of a well-fitted model.

General Observations

For the low-intensity scenario, we see the overall best estimates for both LYG and Cost from the BNN that is trained on data from sampling strategy 4, with gamma-distributed risk and frequency-based screening. Interestingly, for this sampling method, we see an almost inverse relationship between the epistemic uncertainty and the SMAPE of LYG esimates. While the uncertainty increases with increasing risk, the SMAPE decreases dramatically for higher risk ranges (>5). It should also be noted that observations are very sparse above that threshold, with 97% of observations having lower risk. For Cost, the SMAPE is fairly constant for risk above 2.

While those errors are much higher for sampling strategy 3, with uniformly-distributed risk and frequency-based screening, 0.54 and 0.47 respectively, this is mostly caused by bad estimates for low risk value.

For the high-intensity scenario, on average the best estimates across the risk range, for both LYG and Cost, are from sampling strategy 2, with gamma-distributed risk and random screening. For LYG and Cost, this results in a SMAPE of 0.14 and 0.056 respectively. For LYG, we see a more sensible relationship between the SMAPE values and epistemic

uncertainty, which both increase with risk.

On average, sampling methods that use Gamma-distributed sampling perform better than the other methods, with SMAPE values across the risk range being about twice as low on average. This does not yet account for the higher weighing of lower risk observations in the final estimate, in which Gamma-based methods are advantageous. Epistemic uncertainty is lowest around the mode of the risk curve, which is found at 0 and increases with risk, where the data is sparser.

For the other sampling methods, which do not have a defined mode, we see that epistemic variance is lowest around the mean of the risk variable in the data (12.5 for sampling method 1 and 3 and 7.5 for sampling method 5).

4.4 Experiment 4: Overall accuracy of Metamodel

In this section, we report the accuracy of estimates which are derived from the metamodel, i.e. discretization of the BNN estimates. In table 7, results are presented for estimates on test set C-T. To reiterate, this dataset contains estimates of the outcomes for a population of 10M people with Gamma-distributed risk, for a low-intensity and a high-intensity screening scenario. The estimates in this section are the results of aggregation of the subpopulation predictions in experiments 3 and will be a reflection of the accuracy of the entire metamodels, trained on the data of each sampling strategy. The t-statistics and F-statistics presented in the table, are obtained by comparing the aggregated predictions of the metamodel, to the universal test set C-T. The first part will focus on mean estimates and the second part on variance estimates.

Sampling method 2, with data with Gamma-distributed risk and frequency-based screening, is clearly the best performer on all scenarios. This was expected for the high-intensity scenarios, as the sampling strategy's data best matches the shape of the test case's data. However, what is more curious, is that it is also the best performer for the test case with low-intensity screening. This is surprising, because the screening programme of this scenario better fits the data of sampling method 4 and we saw better results in experiment 3. All other sampling strategies have higher errors, with SMAPE values between 0.15 and 0.7. This is less surprising as these are uniform risk strategies, while the risk of the population more heavily weights low risk estimates.

We see that for most models, epistemic uncertainty dominates the prediction uncertainty, which is represented by the sum of epistemic, aleatoric and ensemble uncertainty. Relative to the variance in the data, the aleatoric uncertainty, the epistemic uncertainty is the lowest for sampling strategy 2, which also had the best estimates.

For sampling strategy 4, we see that for Cost in a low-intensity scenario, the uncer-

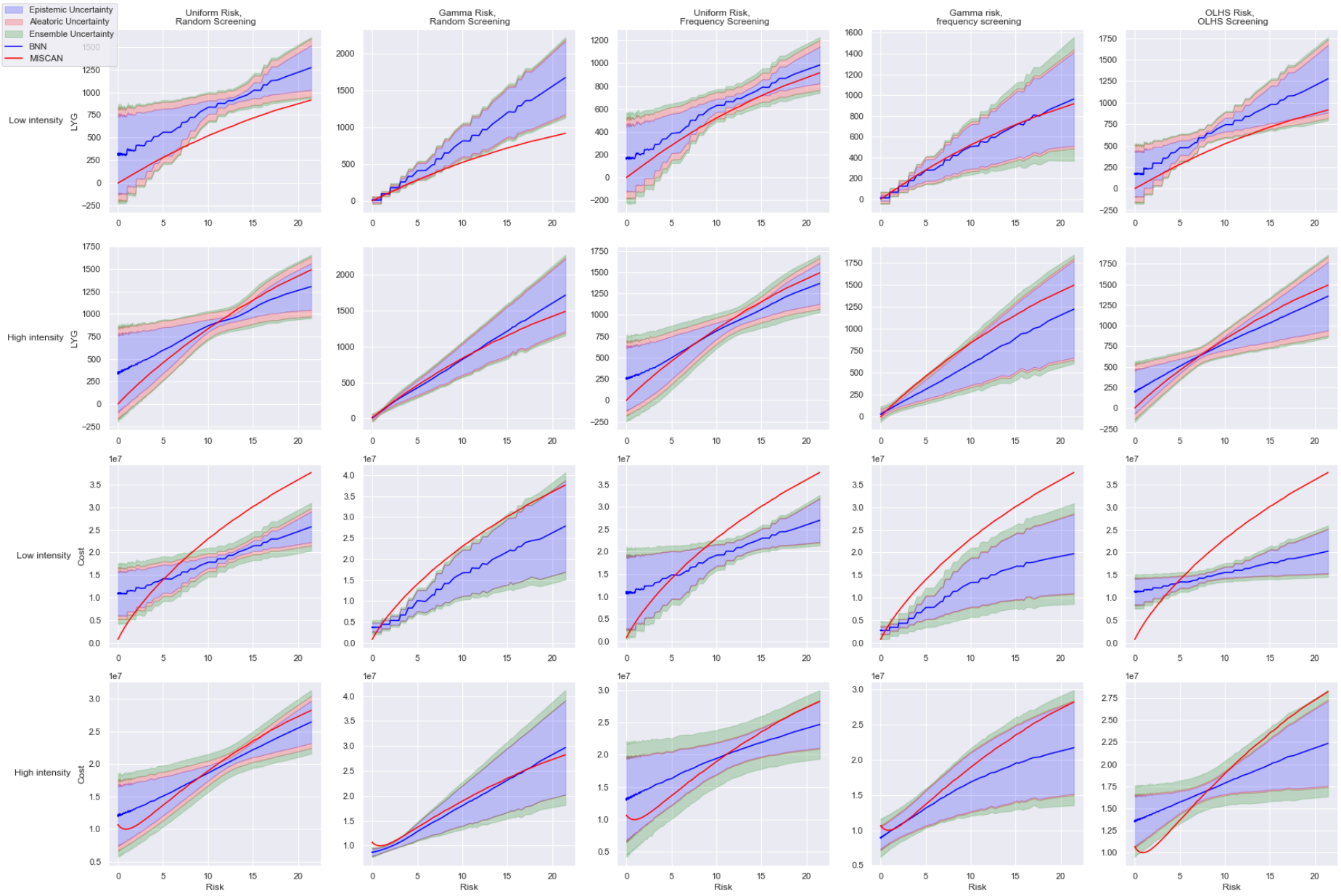


Figure 3: Monte-Carlo Dropout (MCD) BNN estimates of mean, including a border with a width of one standard deviation, split into epistemic, aleatoric and ensemble uncertainty. Plotted against risk for a low and high frequency scenario, for both outcomes, for each sampling strategy.

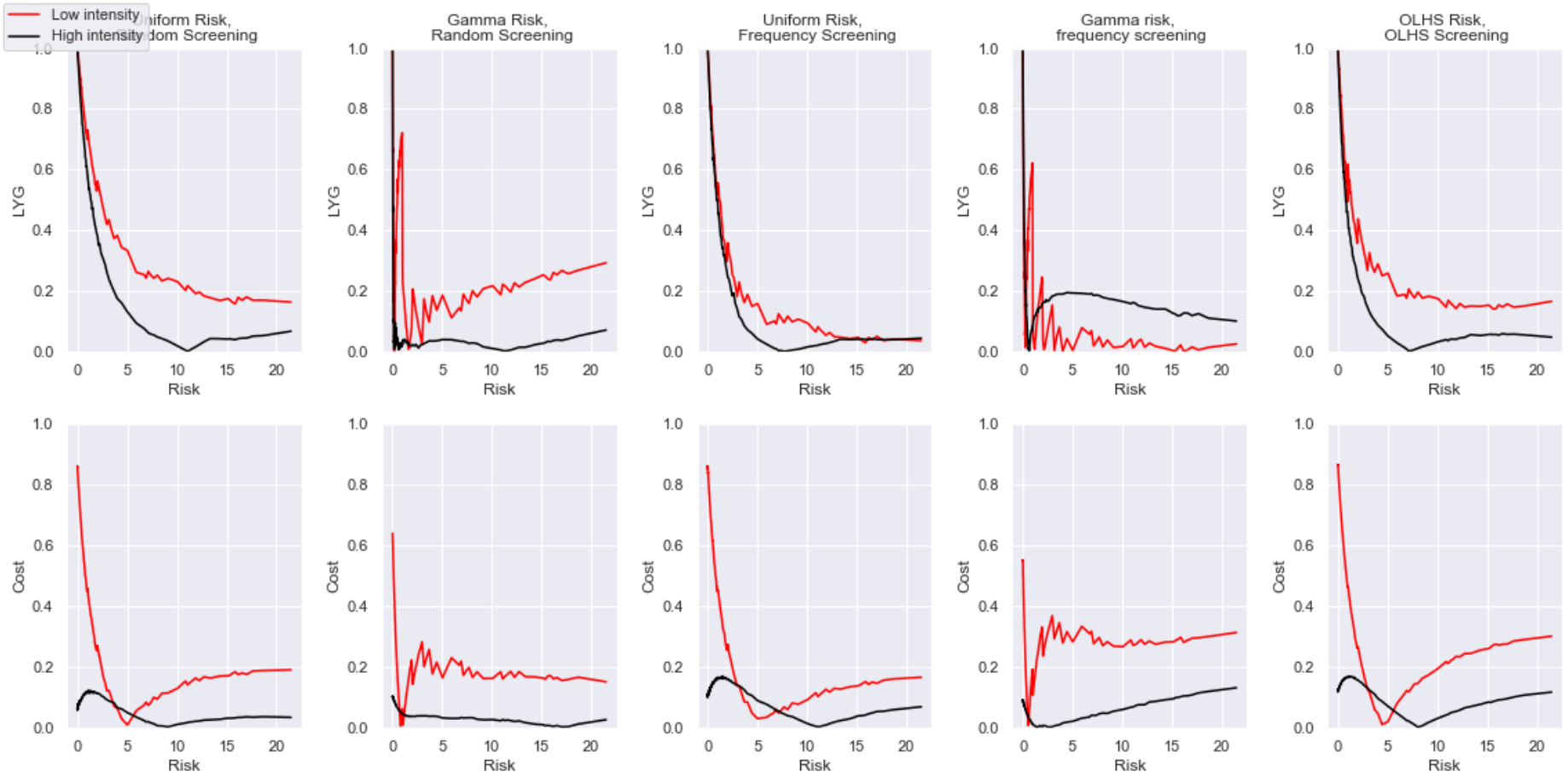


Figure 4: SMAPE error of Monte-Carlo Dropout BNN estimates, plotted against risk for low and high intensity screening scenario, for both outcomes and each sampling strategy

tainty is dominated by ensemble uncertainty, which is an indication that different elements of the ensemble had mean estimates that strongly varied.

Although the SMAPE values for the estimates of sampling strategy 2 seem promising, the t-statistics for the two-sided t-test suggest that we should reject the hypothesis of equal means, for all predictions. Therefore, we conclude that the predictions of the mean by the metamodel are significantly different from the ground truth values generated by MISCAN-Colon.

While prediction errors of strategy 2 are low, this sampling strategy very badly estimates the variance of the scenarios, especially for Cost. F-statistics are obtained by comparing the predicted aleatoric variance to the variance of repeated MISCAN runs (test set C-T). The only non-significant F-value is given by sampling strategy 4, for Cost in a low-screening scenario. Given that this is only 1 out of 20, this would not exceed a false positive correction of 5%.

4.5 Run time

Approximate run times per experiment are provided in table 8. These are the recorded run times for running a single sampling strategy. A modeler's decisions, such as ANN architecture, may have a large influence on the total run time, even when hardware performs constantly. Therefore, the numbers presented here should only serve as an indication of the order of magnitude of the run time.

We can divide the run time into two phases: model development, which covers the data generation and experiments 1 and 2; and model prediction, which covers experiments 3 and 4. Model development entails finding the metamodel hyperparameters that best fit the data of the modeler's purpose. If successful, this step only has to be performed once. The model prediction has to be repeated for each new scenario of interest. It should be noted that specialized, high-powered hardware (Argonne's BEBOP super computer) was used to generate very large datasets. However, due to computational bottlenecks in GPU capacity, most of the time, these were substituted by relatively small datasets of size $1e5$, which can tractably generated with non-specialized hardware. Approximate run times for non-specialized hardware can be found in table 16.

In summary, it takes 0.5 to 2 days to train a new metamodel. Then, it takes approximately 60 to 75 minutes to train an ensemble to make predictions for each screening scenario in experiment 3 and between 0.005 and 0.01 seconds to create an aggregated estimate for experiment 4. Given the design aim of 150,000 predictions, this would take between 5,000 and 6,250 days of constant GPU running, which can be considered intractable for research purposes, especially because it's more costly than running MISCAN-Colon.

Source	Var	Screening	$SMAPE_{\mu}$	t	p	$SMAPE_{var}$	F	p	% epi	% ale	% ens
1: Uniform Risk, Random Screening	LYG	Low	0.720	54433.881	1.000	0.903	19.641	0.000	0.965	0.032	0.003
		High	0.606	35533.635	1.000	0.780	8.077	0.000	0.964	0.034	0.002
	Cost	Low	0.492	43399.787	1.000	**0.310**	1.900	0.000	0.933	0.024	0.043
		High	0.156	34627.933	1.000	0.708	5.848	0.000	0.931	0.028	0.041
2: Gamma Risk, Random Screening	LYG	Low	**0.063**	1413.837	1.000	**0.179**	1.436	0.000	0.722	0.238	0.040
		High	**0.014**	-328.826	0.000	0.257	0.591	1.000	0.725	0.212	0.063
	Cost	Low	**0.059**	2819.409	1.000	0.990	0.005	1.000	0.843	0.001	0.156
		High	**0.004**	753.214	1.000	0.968	0.016	1.000	0.957	0.002	0.040
3: Uniform Risk, Frequency Screening	LYG	Low	0.552	26096.896	1.000	0.845	11.932	0.000	0.939	0.041	0.020
		High	0.509	23914.795	1.000	0.661	4.907	0.000	0.941	0.026	0.033
	Cost	Low	0.493	43604.603	1.000	0.895	0.055	1.000	0.961	0.000	0.039
		High	0.199	46534.819	1.000	0.709	0.170	1.000	0.881	0.000	0.119
4: Gamma risk, frequency screening	LYG	Low	0.071	-1410.111	0.000	0.372	2.185	0.000	0.666	0.311	0.023
		High	0.057	-1251.560	0.000	**0.053**	0.899	0.954	0.555	0.197	0.248
	Cost	Low	0.065	-2759.356	0.000	0.844	0.085	1.000	0.442	0.012	0.547
		High	0.028	5353.241	1.000	**0.587**	0.260	1.000	0.803	0.007	0.190
5: OLHS Risk, OLHS Screening	LYG	Low	0.579	29174.246	1.000	0.879	15.538	0.000	0.928	0.069	0.003
		High	0.461	19719.344	1.000	0.729	6.390	0.000	0.914	0.074	0.012
	Cost	Low	0.501	45035.447	1.000	0.874	0.067	1.000	0.932	0.003	0.066
		High	0.205	48264.017	1.000	0.657	0.207	1.000	0.868	0.003	0.130

Table 7: Accuracy descriptives of the metamodel, for different sampling strategies. The best SMAPE results per scenario are presented in boldface.

Exp.	Unit	Number of Units	Hardware used	Lower (s)	Upper (s)	Total Lower	Total Upper
Data	Datapoints	1×10^5	Bebop (1438 cores)	1×10^{-2}	1×10^{-2}	1×10^3	1×10^3
1	ANNs	168	NVIDIA Quadro M1200 GPU	1.2×10^2	1.8×10^2	2.0×10^4	3.0×10^4
2	BO iterations	300	Tesla K80 GPU	1×10^2	5×10^2	3×10^4	1.5×10^5
Training Total						5×10^4	1.8×10^5
3	Ensemble members	5	Tesla K80 GPU	6×10^2	9×10^2	3×10^3	4.5×10^3
4	Stratification Scenarios	1	Intel Core i7-7701 HQ CPU	5×10^{-3}	1×10^{-2}	5×10^{-3}	1×10^{-2}
Prediction Total						3×10^3	4.5×10^3
Total						5×10^4	2×10^5

Table 8: Approximate run times and hardware used for different experiments, for a single sampling strategy.

5 Discussion

In this section, we first discuss the results per section. Then, we discuss limitations, implications and avenues for future research. First, we reflect on the results of experiments 1 through 4, in section 5.1. Then, with knowledge of the results, we reflect on the design requirements set up in section 3.1, this is presented in section 5.2. These evaluations are summarized, resulting in user recommendations and general contributions to the literature, in section 5.3. Then, limitations of the thesis are presented in section 5.4. Finally, avenues for future research are discussed in section 5.5.

5.1 Discussion of Results

This section will comment on the separate results of the four experiments of this thesis. An overall evaluation of the research and implications for users can be found in section 5.3.

Experiment 1: ANN Hyperparameter Optimization

Because of limitations in computing power, some boundaries needed to be set on the number of ANNs to evaluate. Because this is an a priori decision, this strategy runs the risk of excluding the evaluation of ANN architectures which might be optimal, for example those networks deeper than 5 layers. Based on the results, this seems less of a concern, because we found that the ANN architectures that were tested seemed fairly interchangeable. If only the ANNs at the edges of the range would be optimal, this would suggest the need for a higher range, but this is not the case.

For all sampling methods, it seems to be optimal for both outcomes to train on only one outcome, so not on split cost and not on LYG and cost at the same time. Because

the reported MSE is a generalization error, the fact that NNs with split cost perform bad, could be an indication of those networks overfitting. Another consistent result can be found in the fact that Batch Normalization seems required to balance deeper Neural Networks (of 3 or 4 layers) and not always for more shallow designs (1 or 2 layers). This is not surprising, as deeper networks are prone to overfit and we have mentioned before that Batch Normalization helps in regularization.

The only consistency in the top 10 results for all sampling methods was that it was best to train on a single outcome, so not on both Cost and LYG. This was surprising, since we found that LYG and Cost are correlated. This cannot be attributed to a scaling issue, because the data was normalized for models trained on Both outcomes. It was also consistently optimal to not split the cost. This may be an issue of overfitting due to too many parameters, even though the models were regularized through BN or Dropout.

It would be interesting to see how a metamodel with just the ANN estimates would fare, so without the BNN extension. It does not have to sacrifice mean estimate accuracy for variance estimate accurracy and would be much faster in inference than a BNN. Epistemic uncertainty may then be calculated through bootstrapping, with a separately trained ANN for each iteration.

In conclusion, researchers are encouraged to try different architectures, but should not feel forced to attempt to do this too exhaustively. In the results section, it was shown that ANNs with different amounts of hidden nodes, with or without Batch Normalization may deliver optimal results. Using ANNs for prediction makes for a simplistic setup, but for rather difficult tuning. ANNs are difficult to interpret, which hampers the development of improvement steps. Although ANNs are often considered to be highly apt at fitting high-dimensional functions; for lack of benchmark, we cannot objectively say that they are the best model for our general purpose.

Experiment 2: BNN Hyperparameter Optimization

For almost all sampling methods, the algorithm was not able to converge to a single optimal hyperparameter set, in a way that would suggest a global optimum. Rather, the best for the precision parameter τ in terms of log-likelihood were often far apart, as compared to lower scoring iterations. While multi-modality was expected and is not an issue for the other parameters, it is undesirable, for the precision parameter, which is the sole estimator of MISCAN-Colon's aleatoric uncertainty. An important reason this occurred would be the decision to resample a new dataset for each iteration, which would imply a selection bias.

A weakness of the current setup is the homoscedastic aleatoric uncertainty estimator. It estimates a precision parameter based on the training dataset, while we know that not all training datasets have the same aleatoric variance as the universal test set. It would be

interesting to see what a BNN with heteroscedastic aleatoric uncertainty could achieve in terms of precision estimates and final metamodel prediction of variance.

Bayesian Optimization of the BNN hyperparameters allowed for a straightforward way to find estimates on a continuous scale. An obvious downside to the approach is that the resulting model is optimized for estimates of variance and not necessarily for estimates of the mean. Modelers struggling with the Monte-Carlo Dropout approach of BNN esimation may be interested to try deep ensembles (Lakshminarayanan et al., 2017). In combination with bootstrapping, deep ensemble allow for an easily scalable way to estimate aleatoric and epistemic uncertainty.

Experiment 3: Individual BNN estimates

As expected, all models have the lowest epistemic uncertainty for the risk range in which they have the most data. It was also to be expected that high-intensity screening data performed well on the high-intensity validation sets and vice versa. What was more striking, is that no sampling method seems to be very good at estimating Cost in a low-intensity scenario, with the actual values falling far outside the prediction uncertainty boundary of one standard deviation.

While we have no benchmark of how large the epistemic uncertainty should be, it is striking to see that it is often several times larger than the aleatoric uncertainty. Most of the time, it is minimal at the mean of the risk distribution and that predictions further from the mean are less accurate. This suggests that the model may have minimized the MSE by just predicting the mean, which would be an indication that it was not able to incorporate a more complex mapping, perhaps due to an ill fit.

By looking at the trends, it becomes apparent that the epistemic uncertainty often increases and decreases with the absolute error. According to Scalia, Grambow, Pernici, Li, and Green (2019), this is an indication of meaningful uncertainty, as the model would become more accurate when the most uncertain predictions are removed.

The results of sampling strategy 3, with uniform risk and frequency-based screening are quite disappointing for the prediction of Cost in a low screening scenario. The model seems ill fitted, which is striking, as the low screening scenario is very likely present in the training data of sampling strategy 3. And, if not, at least very similar scenarios. Again, this may be an indication of an ill fit.

Being able to estimate both aleatoric and epistemic uncertainty is a clear strength of BNNs. This allows researchers to formulate a distribution for the prediction uncertainty and allows modelers to look for strengths and weaknesses of the model. Practically, modeling of the epistemic uncertainty showed that the models are not very balanced across the risk range, with epistemic uncertainty being the lowest around the mean of the sampling

distribution. This begs the question whether a weighted average of models trained on different data sets may allow for a more balanced model. For example, a prediction from a model trained on data of sampling strategy 2 may receive a higher weight for low-risk sub-populations and a model based on sampling strategy 3 for high-risk sub-populations. In the same way, models may be combined based on the screening intensity of the scnenario of interest.

Experiment 4: Metamodel final results

Unfortunately, the metamodel performs poorly in the prediction of the variance of the main model. Only the estimates of the variance of LYG for a high-intensity screening scenario yield a p-value for the F-test that is not within the two-sided rejection region. However, with 20 tests conducted, it is hard to reject the notion of a single false positive, at a rate of 5%.

While sampling methods 1 and 5 practically should have approximately the same data for large enough N, the orthogonal methods performed better, although still poorly. This could be an indication that for the size of our dataset, Orthogonal Latin Hypercube methods still have an advantage, perhaps due to less correlation in observations.

Generally, the methods with Gamma-sampled risk outperformed the methods with Uniformly-sampled risk. The only way that the strategies with uniform risk could have performed better, was if model fit and data sufficiency were optimal, such that they would perform similarly in the low risk scenario. Logically, they would then outperform the gamma strategies in terms of high risk scenarios. This is especially desirable to make predictions on stratified scenarios, in which some strata would contain only high risk individuals.

In conclusion, the metamodel trained on sampling method 2 seems promising. Based on t-tests and F-tests, it gives estimates that are statistically significantly different from MISCAN-Colon, which suggests that the metamodel cannot yet replace MISCAN in a research context. However, with SMAPE values ranging between 0.06 and 0.004, it might still be accurate enough to give an indication of promising screening scenarios. The subselection of evaluated scenarios that seem promising, may then be verified by MISCAN-Colon.

5.2 Reflection on the Design Requirements

In section 3.1, requirements for the design were presented. A reflection on these requirements will follow now.

Speed

Timing measurements showed that training a metamodel is a feasible task for any research operation with access to a mid-range GPU. However, to use it for prediction is currently slower than through MISCAN-Colon, assuming a single run of the latter. However, it should be noted that the metamodel essentially replaces a thousand runs of MISCAN-Colon, hence an estimate of the aleatoric uncertainty is included. Another strength of the model is that it predicts on homogeneous sub-populations, building blocks of a heterogeneous population. Hence, if quantile risk predictions are generated for ten screening programmes of interest, which would take under four hours, this allows for the testing of 10^M stratification scenarios, where M is the number of quantile bins. The calculation time of aggregation (experiment 4) allows for thousands of different scenarios per hour. Furthermore, the simple ANNs are able to make a population prediction in seconds, so they could potentially be used for metamodeling efforts, though they lack estimates of uncertainty.

Accuracy

The most extensively evaluated requirement was accuracy. No metamodel was able to make a prediction of the mean that was not statistically different from the mean produced by MICAN-Colon, as measured by means of a two-sided t-test. Nevertheless, results are promising, with sampling strategy 2 producing a metamodel that has SMAPE values around and below 0.05. Some researchers might find this adequate to explore candidate solutions.

Stratification

The metamodel scheme has demonstrated its capacity for stratification well. By estimating on sub-populations, it can be used for a variety of stratification scenarios, without retraining the model.

Balance

Unfortunately, the metamodel is not very well-balanced. It can be seen in figure 4, that SMAPE values are inconsistent and inflate when data is sparser. Especially for very low risk values, the model makes bad predictions on average. This, however, is not surprising, because we have seen that MISCAN-Data is the most variant for very low risk (see figure 13). No statistical measures for balance were employed in this thesis.

Uncertainty

Finally, uncertainty estimation. The estimation of epistemic uncertainty has provided important insights into the strengths and weaknesses of the metamodel. Unfortunately, aleatoric uncertainty estimates were not accurate. Thus, we can conclude that the metamodel has not been able to estimate accurate prediction uncertainty. The uncertainty estimates in this thesis were only evaluated through estimation errors and F-tests. Modelers that are interested in more extensive, quantitative evaluation of the uncertainty, such

as through calibration or dispersion measures, are encouraged to look into the methods proposed by Scalia et al. (2019).

5.3 Conclusions and Contributions

The main aim of this thesis was to develop a metamodel for MISCAN-Colon. Requirements of the model were formulated in terms of speed, balance, accuracy, balance and capacity for stratification and ability to estimate aleatoric and epistemic uncertainty. According to the parameters set for these requirements, the best metamodel derived from the methods of this thesis was not yet up to the standards of scientific research. Its most important strengths were accuracy of mean estimates, its ability to quickly predict various stratification scenarios and its ability to estimate both aleatoric and epistemic uncertainty. Current weaknesses are the balance of its accuracy across the input space, accuracy of uncertainty estimates and its speed in modeling predicting different screening scenarios.

The metamodel is quick, but inaccurate when in terms of stratification. However, it is slow, yet accurate when evaluating different screening scenarios. This limits its usefulness in public health research, as un-stratified scenarios are quicker evaluated by running the main model, MISCAN-Colon and predictions on stratified scenarios are potentially inaccurate, due to the in-balanced accuracy of the model. Nevertheless, if the balance of the accuracy is improved, the metamodel may make for large gains in efficiency, by being able to quickly predict different stratification scenarios.

In conclusion, the general framework of the model holds interesting properties and could potentially be useful for CRC researchers. However, in its current form, it is not fast or accurate enough, so it will require an effort of tuning the model to make it practically useful. It should be noted that if prediction uncertainty is estimated correctly, the model can not just replace a single run of MISCAN, but the high amount of runs required to gain an uncertainty estimate of the main model. Therefore, with adequate accuracy, the metamodel will be useful, even if the inference speed remains the same.

Other than developing a metamodel as a product, this research has provided some insights to advise future metamodelers. First, it was shown that ANN architectures were fairly interchangeable for predictions on MISCAN-Colon sub-populations, as long as they were trained on a single outcome. Other than that, we showed that training on sampling strategy 2, with Gamma-distributed risk and random screening, resulted in the most accurate estimates of the mean.

5.4 Limitations

In experiment 2, the BNN was optimized for variance estimation, not for accuracy. This may cause that estimates from the BNN are less accurate than those of the original ANNs, which are purely optimized for mean estimation. Furthermore, the modeling process assumes that optimal architecture for regular NN estimates will also be optimal for a Monte-

Carlo Dropout architecture. Potentially, there might be architectures more suitable for variance estimation. However, it was deemed too expensive to optimize NN architecture and BNN hyperparameters simultaneously.

The ANNs architectures were trained with log-transformed data, because without log transformation, training and inference seemed intractable for such a large amount of networks. However, to calculate the final MSE error values, which were used to select the optimal architectures, the ANN's predictions were first exponentiated, before being compared to the training data.

Through Google Colab, a free cloud-based service, we had access to considerable GPU computing power. In some cases, for example with patient-sensitive data, researchers may not be able to use such cloud-based services and therefore have limited GPU power. In such a case, it is not feasible to use Neural Networks for training or prediction and researchers are advised to use a different Machine Learning Framework, such as Kriging or Random Forests.

In its current form, speed is still an issue. While, with GPU access, Neural Network inference is much faster than without, hyperparameter optimization and Monte-Carlo inference of estimates is a time-consuming process, as it takes several hundred training and inference iterations until usable results are obtained, which might take days.

In this thesis, no variable selection was employed. While we consider it a strength that the model is able model any test at any age, variable selection might increase accuracy of the model.

No dimension reduction was used. Theoretically, ANNs should be able to handle the dimensionality in this thesis and this high dimensionality was an important reason for the choice of ANNs. However, since accuracy is not up to par yet, dimension reduction might improve generalizing ability and training speed. Furthermore, it opens the door for other methods, such as Random Forests or Kriging.

Finally, data from MISCAN-Colon was used without cleaning. The data is simulated and therefore contains no measuring error. Any outlier in the data is inherent to the statistical mechanics of the work (i.e. informative outliers).

5.5 Future research

In this work, we were only interested in homoscedastic uncertainty, because we were only interested in the variance of the final outcomes. However, modelling heteroscedastic uncertainty might make for more accurate estimates of the aleatoric uncertainty.

In this model we have not employed variable selection. This was done because we want

to be able to model a full screening policy, to measure its outcomes. In further modeling efforts, research may try their hand at variable selection to see if similar outcomes may be achieved through fewer variables.

Metamodeling efforts in this thesis were focused on achieving accurracy through complexity. This complexity may have led to poor generalizability. We chose not to employ dimension reduction of the variables, as to not add an additional layer of abstraction and hamper interpretability. However, this has severely limited our choice of metamodel architectures, such as Kriging. It may be interesting to employ dimension reduction, such as PCA or Moving Least Squares Projection. Alternatively, we may use high-level screening variables, such as the start age, stop age and frequency of tests as input variables, in addition to the risk.

The metamodel has not yet been tested for a stratified population, while, theoretically, it is one of its strengths. If the BNN of sampling method 2 now has to estimate outcomes for a high-risk only stratum, this might cause sampling method 2 to appear less optimal, because it is less accurate in these ranges.

6 Sensitivity

In this thesis, a sub-population size of 1×10^3 and a sample size of 1×10^5 was used to train the metamodels for different sampling strategies. Because of computational constraints, it was not possible to consider different levels for these values, for all sampling strategies. After obtaining results, it was concluded that sampling strategy 2 yielded the best metamodel, hence the aforementioned parameters may be put in perspective. So, this section provides results for a smaller and larger sub-population size, namely 1×10^2 and 1×10^4, with sample size of 1×10^5. Furthermore, a smaller sample size of 1×10^4 is modeled, with a sub-population size of 1×10^3. A larger sample size than the benchmark model was not considered, mostly due to GPU constraints in experiment 3. Similar to the process shown above, data is generated for these alternative scenarios, ANN architectures and BNN hyperparameters are tuned to the data, as described in the Methods section. In short, the metamodels are completely retrained. The performance of these alternative metamodels is evaluated through the same standards as the models in section 4.

The sub-population size determines how large the homogeneous sub-populations are that are run through MISCAN-Colon. As explained before, smaller sub-populations imply a smoother approximation of the outcome curve but noisier data, while larger sub-populations result in the opposite. The results for this analysis are presented in table 9. A smaller sample size should logically lead to worse estimates, as less information is used to train the model.

6.1 Speed

Both in terms of generating training data and conducting experiments to find metamodels, run times are approximately the same for the three different sub-population times. As can be seen in table 16, generating sub-populations of size 1×10^4 takes about 50% longer than training a sub-population of size 1×10^2. Once training data has been generated, the datasets are the same size and the time required to develop the metamodel does not differ across sub-population sizes.
A smaller sample size uses only 10% of the data and therefore approximately 10% of the time to generate it. While the gains in terms of speed were not measured for all experiments separately, the entire process, including experiments 1-4 required about a third of the time.

6.2 Accuracy

In table 9, predictions on universal test set C-T are found, similar to those in table 5, in the results section. For estimates of the mean, the model trained on the largest sub-populations, outperforms the other models for three out of four scenarios. The model trained on sub-populations of size 1×10^3 only outperforms it for estimates on LYG, with low intensity screening, where it is very close to the predictions of the larger sub-populations. In terms of variance estimates, we see the opposite: the model trained on the smallest sub-populations has the best estimates, with only the benchmark model perform-

ing better when predicting LYG, with low intensity screening.
The metamodel trained on a smaller sample size actually makes better predictions for the mean in a high intensity LYG and low intensity Cost scenario, than the benchmark. This is surprising, as the model has seen less data. This is an indication that the benchmark model might not be improved with more data, but with better tuning. However, a SMAPE value of of 0.137 for prediction of LYG in a high intensity screening scenario may be too high for researchers, even when using the metamodel for exploration. In figure 7, low intensity predictions are very similar to the benchmark model, while high intensity cost predictions are, surprisingly, more balanced.

6.3 Balance

Figure 5 portrays the SMAPE values for predictions across the risk spectrum. For all models, we see similarly trending behavior. Considering that the metamodels used to generate the predictions for these SMAPE results only share a main model for data generation and test set, not training data, ANN parameters or BNN parameters, the behavior between the datasets is strikingly similar. Only for Cost predictions, with high intensity screening, only the benchmark model seems to suffer from inaccurate estimates for low risk data.

6.4 Uncertainty

Epistemic uncertainty values across the risk spectrum may indicate how certain a model is about its estimates for different regions of the solution space. These values are found in figure 6. We see that epistemic uncertainty is the highest for the smallest sub-populations and the smallest for the benchmark model, for LYG predictions. Remarkably, for Cost estimates, epistemic uncertainty is the lowest for the largest sub-populations and highest for the benchmark model. Other than a difference in magnitude, the curves are very similar in shape.
In figure 8, it can be seen that epistemic uncertainty is lower for larger sample size when predicting LYG and higher when predicting Cost.

6.5 Conclusion

When looking at sub-population size in this section, we face a bias-variance trade-off, in terms of prediction accuracy. Estimating smaller sub-populations makes for metamodels that better fit the variance of the DGP, while larger sub-populations result in better predictions of the mean. Other than that, the different metamodels behave very similarly and it is up to the modeler's preferences which sub-population size they prefer.
Surprisingly, using more data did not unequivocally imply better results. Both in terms of uncertainty reduction and prediction accuracy, there is no clearly preferred sample size. In terms of speed, a smaller samplesize is preferred.

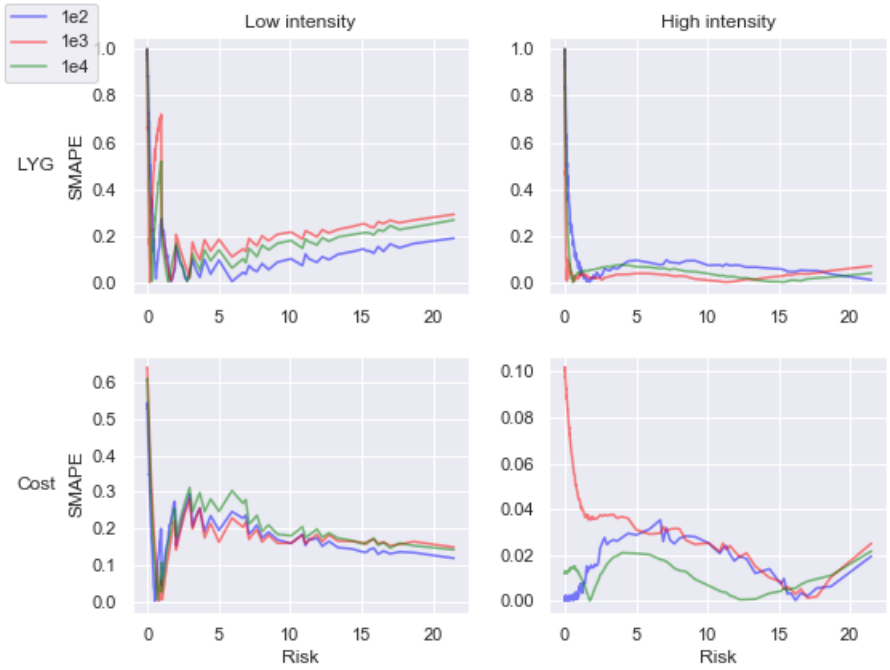


Figure 5: SMAPE values for predictions on universal test set C-T, for models trained using sampling strategy 2, on sub-population sizes of $1 \times 10^2, 1 \times 10^3$ and 1×10^4 and sample size 1×10^5

Sub-pop Size	Sample Size	Var	Screening	$SMAPE_{\mu}$	$SMAPE_{var}$
1×10^2	1×10^5	LYG	Low	1.41×10^{-1}	3.66×10^{-1}
			High	8.20×10^{-2}	6.02×10^{-2} ^{o}
		Cost	Low	3.56×10^{-2}	5.98×10^{-1}
			High	6.49×10^{-3}	1.27×10^{-1} ^{o}
1×10^3	1×10^5	LYG	Low	6.31×10^{-2} ^{o}	1.79×10^{-1} ^{o}
			High	1.43×10^{-2}	2.57×10^{-1}
		Cost	Low	5.89×10^{-2}	9.90×10^{-1}
			High	4.01×10^{-3}	9.68×10^{-1}
1×10^4	1×10^5	LYG	Low	6.86×10^{-2}	2.39×10^{-1}
			High	3.46×10^{-3} ^{o}	5.96×10^{-1}
		Cost	Low	1.15×10^{-2} ^{o}	8.51×10^{-1}
			High	6.33×10^{-3} ^{o}	6.02×10^{-1}
1×10^3	1×10^4	LYG	Low	1.37×10^{-1}	4.39×10^{-1}
			High	6.56×10^{-3}	7.24×10^{-1}
		Cost	Low	5.18×10^{-2}	2.57×10^{-1} ^{o}
			High	2.72×10^{-2}	6.78×10^{-1}

Table 9: Descriptive Statistics for predictions on universal test set C-T, by metamodels trained on data using sampling strategy 2, with sub-populations of size 1×10^2, 1×10^3 and 1×10^4. The best SMAPE per scenario is marked by an ^{o}

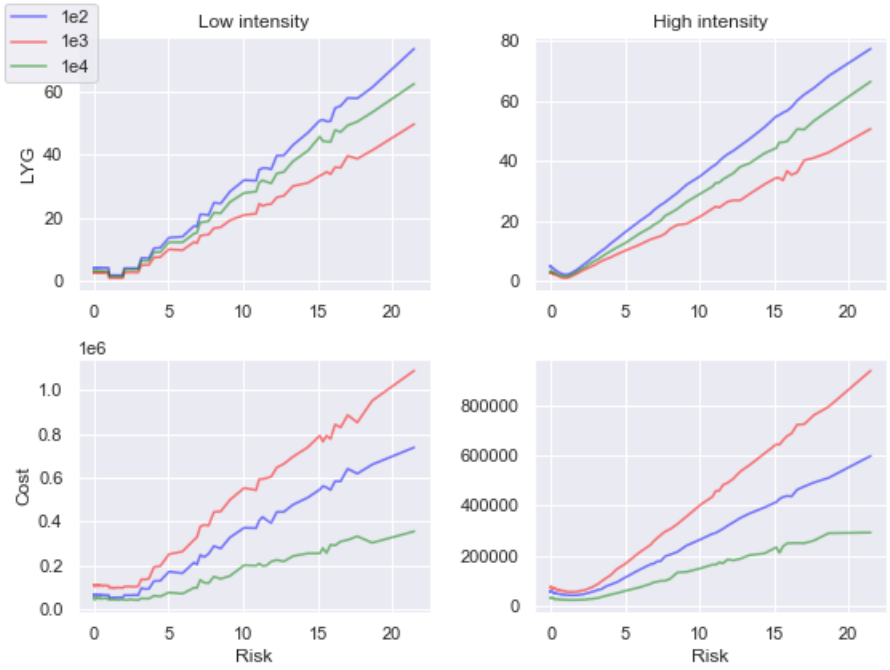


Figure 6: Standard deviation values for epistemic uncertainty of predictions on universal test set C-T, for models trained using sampling strategy 2, on sub-population sizes of $1 \times 10^2, 1 \times 10^3$ and 1×10^4 and sample size 1×10^5

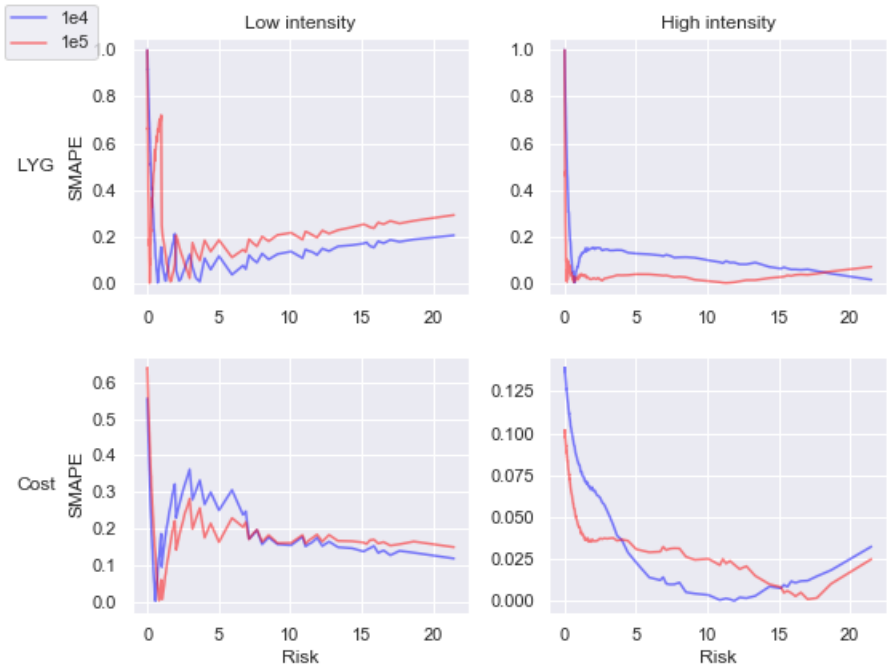


Figure 7: SMAPE values for predictions on universal test set C-T, for models trained using sampling strategy 2, with sub-population size 1×10^3 and sample sizes 1×10^4 and 1×10^5

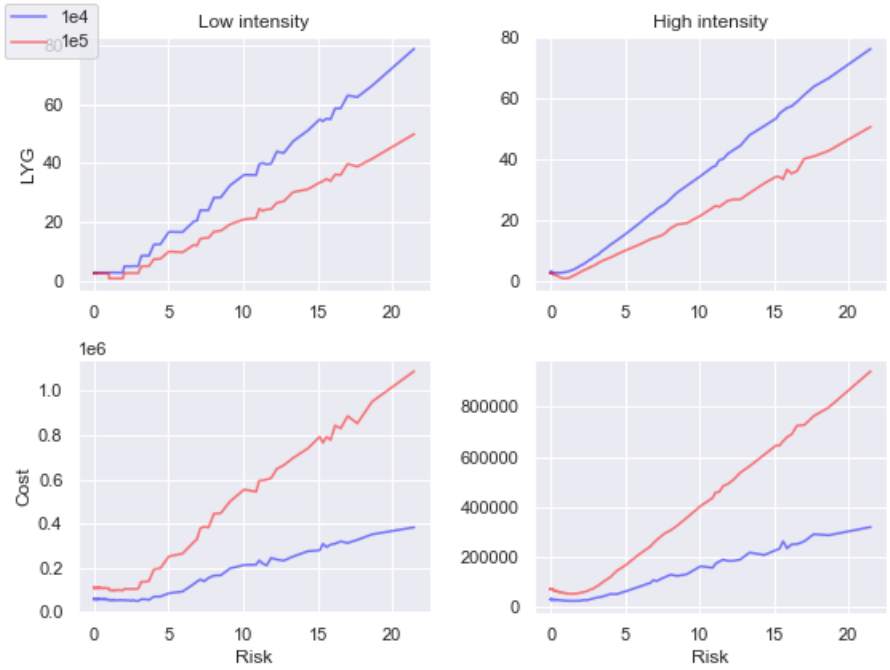


Figure 8: Standard deviation values for epistemic uncertainty of predictions on universal test set C-T, for models trained using sampling strategy 2, with sub-population size 1×10^3 and sample sizes 1×10^4 and 1×10^5

Appendix

A Parameter Descriptions

Parameter	Interpretation
α	Stepsize of Adam optimizer
β_1, β_2	Exponential decay rates for Adam optimizer
θ	Scale parameter for Gamma function
κ	Shape parameter for Gamma function
τ	Model Precision
Ω	end greek
a	Activation value in ANNs
$c(a)$	Activation function in ANNs
C_i	Cost outcome for sub-population i
M	Stratum size
$L(W)$	Loss function of weight matrix W
L_i	Life years gained outcome sub-population i
N	Sample size
p	Dropout Probability in ANNs
R	Individual Risk
S	Screening
W	Weight Matrix
X	Matrix of independent variables
Y	Matrix of dependent variables

Scenario	μ	$\hat{\mu}$	SMAPE $_{\mu}$	σ^2	$\hat{\sigma}^2$	$F(p)$
LYG, Low	5.665×10^5	5.654×10^5	9.580×10^{-4}	2.860×10^6	2.838×10^6	1.001 (0.453)
LYG, High	9.631×10^5	9.727×10^5	4.931×10^{-3}	6.955×10^6	7.277×10^6	0.956 (0.763)
Cost, Low	3.873×10^{10}	3.873×10^{10}	1.742×10^{-6}	2.978×10^{15}	2.535×10^{10}	1.175 (0.00547)
Cost, High	9.233×10^{10}	1.060×10^{11}	6.898×10^{-2}	9.677×10^{14}	1.012×10^{15}	0.9562 (0.760)

Table 11: Discretization Results

B Validation of Discretization scheme

In this section, we present the accuracy of the discretization process, in table 11. The aim of this section, is to show that if the sub-populations are estimated correctly, the discretization correctly calculates outcomes for a heterogeneous population. The ground truth for both the homogeneous sub-populations and the heterogeneous populations are obtained by running MISCAN-Colon 1,000 times for these scenarios.

For each validation scenario, LYG and Cost for low-intensity and high-intensity screening, we compare the results. We observe that for estimations of LYG, results are satisfactory, for both low and high intensity screening. The prediction percentage error is low and the F-statistic for variance comparison is not significant, so we cannot reject the null hypothesis that they belong to the same distribution.

For cost, in the low-screening scenario, we see a very accurate estimate of the mean, but the estimate of the variance is not as accurate. With a two-sided rejection region of 0.05 and a Bonferroni correction for four repeated measurements, we maintain a lower critical value of 0.00625. The p-value is close, but significant, therefore we reject the null-hypothesis that they come from the same distribution. For Cost, the F-statistic is not significant, but the SMAPE for the mean estimate value is relatively high.

C Choosing a metamodel framework

A well-known way of defining the objective function to optimize, is through *Radial Basis Functions (RBF)* (Sóbester et al., 2014). RBF is a local smoothing method, in which a linear combination of a series of basis functions is used to approximate the main model. The shape of these functions is pre-determined by the modeler and may vary. A popular choice for the shape is Gaussian, which is effective at estimating prediction errors. Estimation is only based on the radial distance of individual data points to the origin. Like most local smoothing methods, RBF methods are ineffective at estimation at high dimensionality. For these methods, the median distance of the closest point to the origin grows with the dimensionality of the model and population of the input space grows sparser. For both these issues, the amount of data required to train the model grows exponentially with the amount of dimensions in the model (Hastie, Tibshirani, Friedman, & Franklin, 2005).

Another popular meta-modeling technique, especially in (and also originating from) the geostatistical sciences, is Kriging (Freedman, 2009), which uses Gaussian Processes to obtain least-squares estimates of the unknown variable parameters. The technique is used in many areas of research and has the significant benefit of providing uncertainty in its estimates, making it suitable for prediction analysis. However, it has been found that Kriging is not effective at high-dimensional problems either, with computational time growing exponentially with each added variable. This occurs due to the fact that the covariance matrix of independent variables needs to be inverted several times (Bouhlel, Bartoli, Otsmane, & Morlier, 2016). Some work has been done to lift the curse of dimensionality, mainly through dimension reduction (such as Bouhlel et al. (2016), but this would require two additional layers of complexity, one for quantification of categorical variables and one for dimension reduction.

More traditional methods, such as fitting polynomial functions to the data, require heavy assumptions on the shape of the model, do not fare well in high dimensionality and are not equipped to approximate smooth functions (Hussain et al., 2002). *Support-Vector Machines (SVM)* techniques are a promising field of research in metamodeling (Lai et al., 2006).

Two very promising techniques for high-dimensional, numerical/categorical problems, come from the domain of Machine Learning (ML) and are known as *Artificial Neural Network (ANN)s* and *Random Forest (RF)s*. Random Forests are an ensemble technique, which uses a large amount of sparse, high-variance decision trees to fit the model and can be used for both classification and regression (Breiman, 2001). The technique has enjoyed much attention since its conception and is well-known for good performance under high dimensionality and missing data. A downside is that it is not able make inferences outside the range of the training data (Roßbach, 2018). Additionally, RF techniques do not inherently estimate and return a measure of variance. Nevertheless, Mentch and Hooker (2016) are able to infer an asymptotically normal estimator of the variance of predictions of RFs.

Yet, these techniques have sparsely been adopted by the scientific community.
Artificial Neural Network techniques are a very popular technique and are often associated with important buzzwords of the 2010's, such as Artifical Intelligence (AI) and Deep Learning.

D Simulation Study of MISCAN-Colon

In this section, we explore the observed variance in Cost and LYG, for different variations on a base case. To understand the stochastic nature of the main model, MISCAN-Colon, we run the different configurations a high number of times and record the observed mean and variance. We assume several factors may influence variance, such as individual risk and the screening strategy employed, we also vary these to observe the influence on variance, in order to support the choice for a population size.

Setup

We vary Screening, Risk and Population size to generate scenarios, which we run a high number of times.

This serves as a Monte-Carlo estimate of the mean and variance of the model for specific scenarios. According to [table], simulation time does not increase with population size up to 1,000 individuals.

As a trade-off assume 1,000 individuals per simulation as the base case and simulate variations on the base case, to observe the effect on variability. We will use the coefficient of variation, denoted by

$$CV(\mu, \sigma) = \frac{\sigma}{\mu},$$

to observe the relative impact of different variables on the variability. Furthermore, the base case will assume the mean of the risk distribution of [US population], which is assumed to be one, and follow the low-intensity screening scenario, as described in section 3.4.
First, we evaluate the effect of varying population size. Then, we will vary individual risk and finally, we will look at the effect of screening on variability.

We run 10,000 simulations of the base-case, each with unique random seeds, so the Random Number Generator (RNG) will be initiated differently each time. Except for the random seeds, all simulations are identical. Therefore, all variability in outcome can be attributed to the stochastic nature of the model and, from the point of view of the meta-model, can be considered to be data uncertainty, also known as 'Aleatoric Uncertainty' or 'unknown unknowns'. This uncertainty will propagate unchanged to the final predictions of the metamodel, no matter how much data is used to train the model. This can be distinguished from 'Epistemic Uncertainty', or 'known unknowns', which is the uncertainty introduced by an insufficiently trained metamodel and can be reduced by exposing it to more data.

D.1 The Base Case

For the 10,000 runs, LYG has a mean of 57.23 and a standard deviation of 17.11. The variance is 292.63, the average variance per person in LYG is then obtained by dividing the

	Screen	Treat	Diag	Sympt	Comp	Surv	Total Cost	LYG
Mean	5.78×10^5	2.76×10^6	1.94×10^5	1.46×10^4	1.39×10^5	5.83×10^5	4.26×10^6	57.23
Var	1.14×10^8	2.85×10^{11}	1.36×10^8	1.35×10^7	1.53×10^7	6.40×10^8	2.84×10^{11}	292.63
$\frac{\text{Var}}{\text{m}}$	1.14×10^5	2.85×10^8	1.36×10^5	1.35×10^4	1.53×10^4	6.40×10^5	2.84×10^8	0.29
CV	0.01	0.19	0.06	0.25	0.02	0.04	0.12	0.29
ρ_{LYG}	-0.07	0.21	0.13	-0.01	0.05	0.06	0.21	1.0

Table 12: Descriptive statistics for different Cost Measures, per 1,000 individuals

variance per the number of individuals per simulation, resulting in 0.29. The coefficient of variation for LYG in the base case equals 0.29. For Total Cost, the mean is 4.26e6 and the standard deviation is 5.23e5. The variance is 2.84e11, which means the variance per individual is 2.84e8. The coefficient of variation is 0.12.

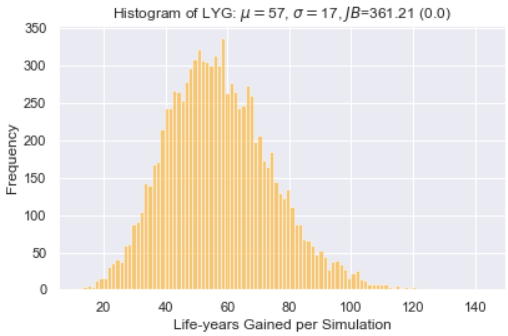
It already becomes apparent that, while Cost has greater absolute variance, this can be attributed to overall higher values and it is more stable throughout iterations of the same scenario than LYG. Histograms of the 10,000 runs can be found in figure 10. It can be observed that the null hypothesis of normality is rejected by the Jarque-Bera statistic for both LYG and Cost.

The correlation between LYG and Total Cost observed in the base case is 0.22 ($p = 0.000$). In figure 10a, we can observe the relationship between the natural logarithm of Cost and LYG and observe a positive correlation. The slope coefficient for the linear regression (black line) equals 0.54 ($p = 0.000$), which suggests that for every 1% increase Cost, LYG increases approximately 0.5%. This linear relationship can be explained by the fact that they are both likely to increase with the prevalence of CRC in the simulation. If we look at the different subcomponents the total cost measures are made up of in figure **??**, we see that the total cost is largely dominated by the treatment cost.

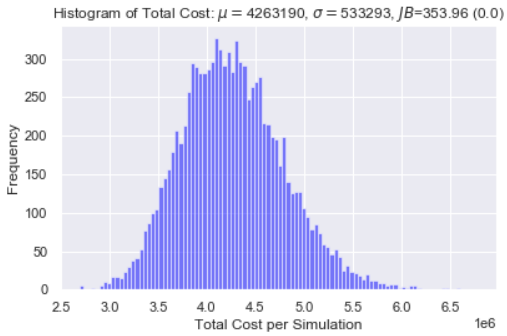
In table 12, we see that Symptomatic Costs are the most variant, based on CV, but Treatment Costs are the greatest contributors to the variance of Total Costs. We see that Treatment Costs are strongly positively correlated with LYG, which can again be explained by their joint relationship with the prevalence of CRC. One can note that the variances do of the cost-subcomponents do not add up to the variance of their sum, the Total Cost, as their covariance is not zero.

D.2 Variance caused by population size

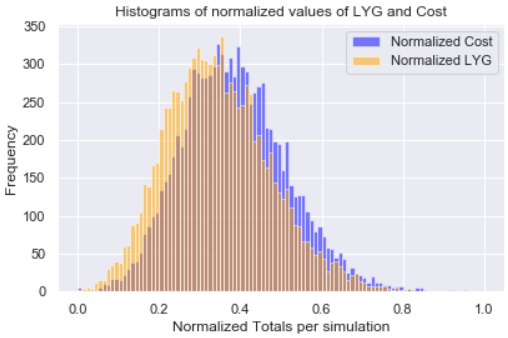
We explore population sizes $[1, 10^1, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7]$ and keep the rest of the base case constant. MISCAN is a micro-simulation model which means that each individual is simulated individually and independently. This is reflected in table 13, where the mean and variance increase linearly with the amount of individuals simulated, m. Because the CV is proportional to the standard deviation and inversely proportional to the mean, it diminishes with a factor $\frac{\sqrt{k}}{k}$, if the population is multiplied by a factor k. Practically, this implies that simulations get more stable with increased populations, which was to be expected.



(a) Histogram of LYG

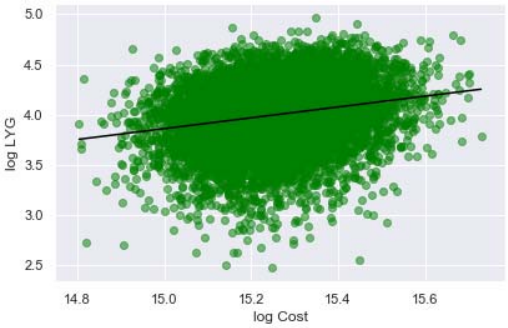


(b) Histogram of Total Cost

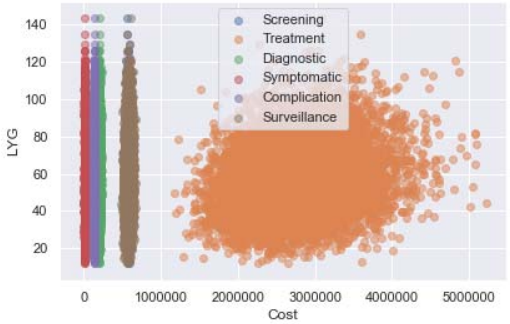


(c) Combined Histogram of normalized LYG and Total Cost

Figure 9: Histograms of Cost, LYG and normalized Cost and LYG



(a) Scatterplot of Total Cost vs. LYG



(b) Scatterplot of different cost subcomponents vs. LYG

Figure 10: Scatterplots of Cost vs. LYG

Pop. size	LYG μ	Var	CV	Cost μ	Var	CV
10^0	5.12×10^{-2}	2.97×10^{-1}	9.82	4.33	3.29	0.41
10^1	0.582×10^{-1}	3.03	2.97	4.34×10^1	3.45×10^1	0.13
10^2	5.58	2.88×10^1	0.96	4.33×10^2	3.32×10^2	0.04
10^3	5.58×10^1	2.84×10^2	0.3	4.33×10^3	3.33×10^3	0.01
10^4	5.58×10^2	2.81×10^3	0.09	4.33×10^4	3.38×10^4	0.00
10^5	5.59×10^3	2.80×10^4	0.02	4.33×10^5	3.38×10^5	0.00
10^6	5.59×10^4	2.80×10^5	0.0	4.33×10^6	33.40×10^6	0.00
10^7	5.59×10^5	2.80×10^6	0.0	4.33×10^7	3.31×10^7	0.00

Table 13: Descriptive statistics for Cost and LYG for different Screening strategies, n=10,000

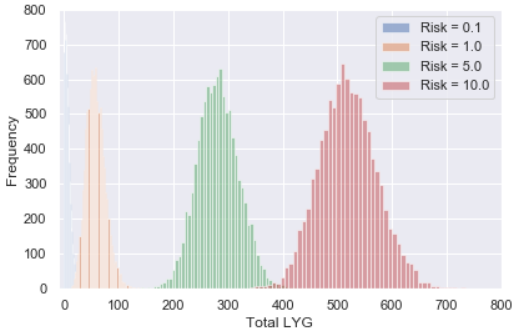
Risk	LYG μ	Var	Var/m	CV	Cost μ	Var	Var/m	CV
0.1	5.54	28.09	0.02	0.95	1.21×10^6	3.08×10^{10}	3.08×10^7	0.144
1.0	57.23	292.63	0.29	0.29	4.26×10^6	2.84×10^{11}	2.84×10^8	0.125
5.0	281.22	1394.0	1.39	0.13	1.40×10^7	1.13×10^{12}	1.13×10^9	0.076
10.0	518.51	2688.5	2.68	0.09	2.29×10^7	1.96×10^{12}	1.96×10^9	0.061

Table 14: Descriptive statistics for Cost and LYG for different risk means, n= 10,000

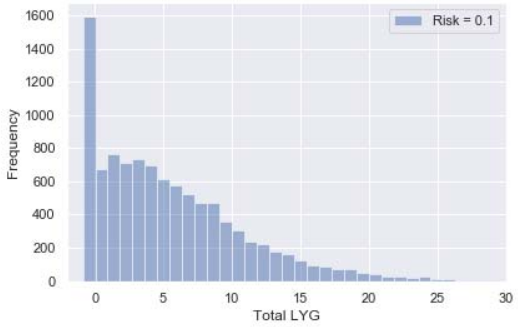
D.3 Variance caused by individual risk

In table 14, we see descriptive statistics for simulations that differ from the base case in terms of individual risk (the base case is printed as boldface). In terms of LYG, the mean expectantly increases with risk, as CRC prevalence increases with risk. Interestingly, we see the coefficient of variation decreasing, which indicates that high risk individuals introduce less uncertainty to the model, in terms of LYG. This can be explained by the fact that there is less uncertainty in the prevalence and therefore LYG. In other words, the more certain we are people get sick, the more certain we are of the lifeyears we can gain. We observe in figure 11b that for low-risk individuals, the mode of LYG is zero observed LYG. In terms of cost, we see a much less steep increase of the mean with risk. This makes sense, as we have found previously that costs are more stable than LYG due to constant costs. Nevertheless, we do see a notable increase in terms of variance and relative variability, especially with very high risk, so very high-risk individuals seem to also introduce more variability in observations of cost. In figure 11c, we observe few surprises, with cost increasing for higher risk, which can be explained by a higher incidence of CRC.

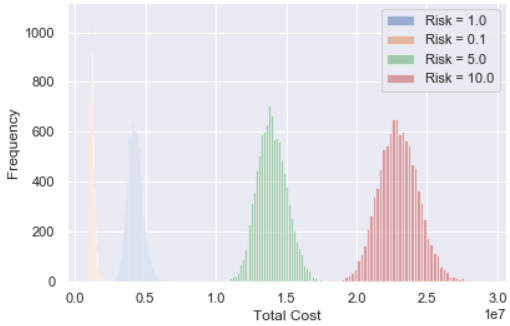
Figure 11 portrays similar results, but with more different observations of risk and. We can see the means of both LYG and Total Cost smoothly increase with risk. For this situation, in which screening is constant, we can fairly accurately fit a simple polynomial function to both LYG ($L = 118.44 * r^{0.69} - 67.44,\ R^2 = 0.999$) and Cost ($C = 5.84 \times 10^6 * r^{0.62} - 1.36 \times 10^6,\ R^2 = 0.999$). For a high-frequency screening scenario, as described in section 3.4, we perform a similar Monte Carlo simulation. We can then fit $L = 165.77 * r^{0.73} - 1.32,\ R^2 = 0.999$ and $C = 1.93 \times 10^6 * r^{0.79} - 6.85 \times 10^6,\ R^2 = 0.998$ Plots of these functions are overlaid in the same figure. This shows that LYG and Cost might be easily



(a) Histograms of LYG for different risks



(b) Histograms of LYG for risk = 0.1



(c) Histograms of Total Cost for different risks

Figure 11: Histograms comparing metrics for different risk measures, n = 10,000

Screening	LYG μ	Var	Var/m	CV	Cost μ	Var	Var/m	CV
None	0.0	0.0	0.0	*Undefined*	3.83×10^6	3.67×10^{11}	3.67×10^8	0.158
Base	57.14	289.09	0.28	0.29	4.26×10^6	2.75×10^{11}	2.75×10^8	0.123
7-yearly Col	88.95	615.96	0.61	0.27	6.30×10^6	1.49×10^{11}	1.49×10^8	0.061
Rand(low)	95.03	721.18	0.72	0.28	9.61×10^6	1.28×10^{11}	1.28×10^8	0.037
Rand(high)	101.0	762.92	0.76	0.27	9.00×10^6	1.04×10^1	1.04×10^8	0.035

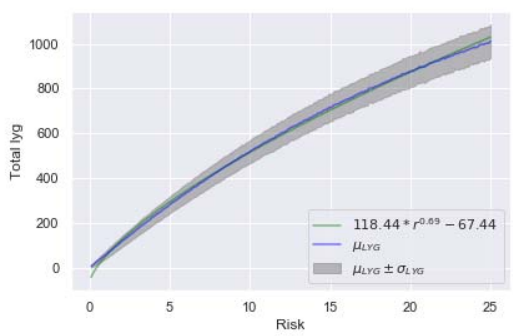
Table 15: Descriptive statistics for Cost and LYG for different Screening strategies, n=10,000

predictable, if screening is kept constant, which might be useful for policy makers.

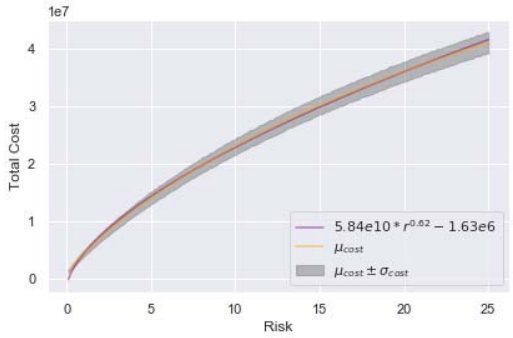
D.4 Variance caused by Screening

In table 15, we see that variability in LYG remains fairly constant, regardless of the intensity of screening. While the amount of LYG increases, the variance does not increase proportionally. In cost, we see that CV decreases with more intense screenign. Total Cost for Random screening is much higher, mostly explained by a large increase in the Screening Costs. Similarly to increasing risk, we can speculate that more intense screening leaves less room for stochasticity. Nevertheless, this does not agree with the constant CV of LYG.

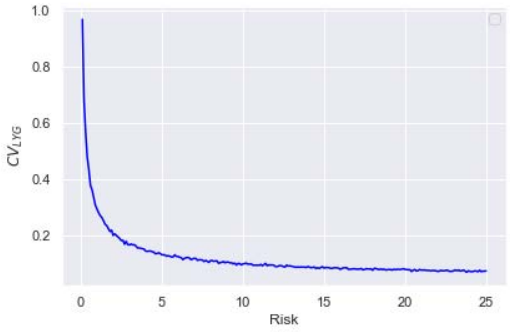
In figure 13b, we see that the costs for 7-yearly col and bi-yearly col are almost identical. This is caused by the fact that more frequent screening (and therefore higher screening



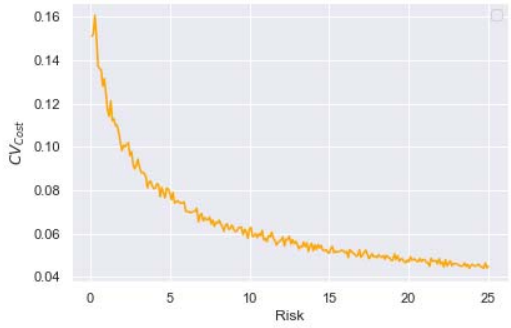
(a) Observed and fitted LYG



(b) Observed and fitted Total Cost



(c) Coefficient of Variation of LYG



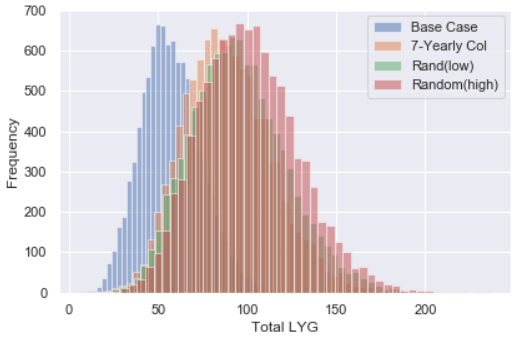
(d) Coefficient of Variation of Total Cost

Figure 12: Plots illustrating the mean and standard deviation of LYG and Total Cost as a function of risk, including least squares fit, n = 1,000 per 0.1 risk increment

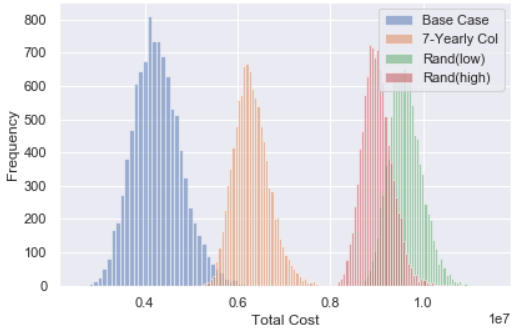
cost) negates the need for surveillance, and is therefore compensated by lower surveillance cost.

D.5 Time required to run MISCAN-Colon

Run-time for MISCAN-Colon is a significant bottleneck for those conducting cost-effectiveness studies. The total run-time of a MISCAN-Colon scenario most importantly depends on the size of the simulated population. We notice that the simulation time for each scenario is comprised of a start-up time, which depends on the type of storage used, and a variable simulation time, which depends on the speed of the processor used and the size of the population. The variable simulation time grows approximately linear with the population size. Therefore, we can formulate the total run time of a MISCAN-Colon scenario as $T = b + d * N$, in which T is the total run time, b is the start-up time and d is the speed coefficient. Tests were conducted on two different computers, one using virtual storage and one using local (SSD) storage. Results are presented in table 16. It can be observed that, due to the constant time required to start up a simulation, there is little time difference between simulating the four lowest levels. After that, the variable time becomes a more important factor in the total time. Therefore, when simulating small populations, large gains can be made by using fast means of storage, such as local SSD, but this advantage diminishes almost completely for larger population sizes, given that processor speed is constant.



(a) Histograms of LYG for different Screening



(b) Histograms of Total Cost for different Screening, Random excluded

Figure 13: Histograms comparing metrics for different screening strategies, n = 10,000

Population size	Virtual Storage	Local Storage
1	9.52	0.53
1×10^1	9.24	0.50
1×10^2	9.24	0.52
1×10^3	9.38	0.55
1×10^4	9.87	0.76
1×10^5	11.8	2.66
1×10^6	30.1	22.3
1×10^7	226	220
$\hat{b}$	9.24	0.50
$\hat{d}$	2.1e-5	2.1e-5

Table 16: Run-time in seconds for simulating a single population with variable population size, assuming one core is used. Based on averaging 100 runs. Parameter estimates are based on linear regression. The computer using virtual storage was using a quadcore Intel Core i7-6700 CPU with 8 GB RAM and the computer using local storage was using a quadcore Intel Core i7-7701HQ CPU with 8 GB RAM.

E Results

E.1 Experiment 1

See table 17 for LYG and table 18 for cost.

No: Risk, Screen	ID	Rank	Layers	Nodes	Cost Split	BN	Y	MSE	SMAPE
1: Unif, Unif	165	0	4	128	No Split	True	LYG	8.53×10^{3}	7.92×10^{-4}
	50	1	1	512	No Split	False	Both	8.84×10^{3}	3.87×10^{-3}
	166	2	4	256	No Split	True	LYG	8.92×10^{3}	1.08×10^{-3}
	49	3	1	256	No Split	False	Both	8.94×10^{3}	3.43×10^{-3}
	164	4	3	512	No Split	True	LYG	9.16×10^{3}	6.39×10^{-3}
	149	5	2	512	No Split	False	LYG	9.53×10^{3}	4.00×10^{-4}
	148	6	2	256	No Split	False	LYG	9.61×10^{3}	6.80×10^{-3}
	144	7	1	128	No Split	False	LYG	1.04×10^{4}	4.57×10^{-4}
	51	8	2	128	No Split	False	Both	1.04×10^{4}	3.40×10^{-3}
	145	9	1	256	No Split	False	LYG	1.05×10^{4}	6.45×10^{-4}
2: Gam, Unif	145	0	1	256	No Split	False	LYG	1.02×10^{3}	6.28×10^{-3}
	146	1	1	512	No Split	False	LYG	1.12×10^{3}	1.88×10^{-2}
	167	2	4	512	No Split	True	LYG	1.12×10^{3}	6.77×10^{-3}
	162	3	3	128	No Split	True	LYG	1.16×10^{3}	1.37×10^{-2}
	164	4	3	512	No Split	True	LYG	1.21×10^{3}	2.93×10^{-2}
	166	5	4	256	No Split	True	LYG	1.24×10^{3}	3.16×10^{-2}
	148	6	2	256	No Split	False	LYG	1.39×10^{3}	1.33×10^{-2}
	68	7	3	512	No Split	True	Both	1.42×10^{3}	2.52×10^{-2}
	117	8	4	128	Categories	True	Both	1.53×10^{3}	2.60×10^{-2}
	165	9	4	128	No Split	True	LYG	1.57×10^{3}	6.86×10^{-3}
3: Unif, Freq	166	0	4	256	No Split	True	LYG	7.30×10^{3}	1.23×10^{-2}
	163	1	3	256	No Split	True	LYG	8.47×10^{3}	1.47×10^{-2}
	162	2	3	128	No Split	True	LYG	8.73×10^{3}	2.46×10^{-2}
	69	3	4	128	No Split	True	Both	9.21×10^{3}	3.35×10^{-2}
	167	4	4	512	No Split	True	LYG	9.29×10^{3}	2.91×10^{-3}
	164	5	3	512	No Split	True	LYG	9.42×10^{3}	1.33×10^{-2}
	68	6	3	512	No Split	True	Both	1.02×10^{4}	2.45×10^{-2}
	146	7	1	512	No Split	False	LYG	1.02×10^{4}	6.22×10^{-3}
	144	8	1	128	No Split	False	LYG	1.03×10^{4}	4.28×10^{-3}
	161	9	2	512	No Split	True	LYG	1.05×10^{4}	9.62×10^{-3}
4: Gam, Freq	165	0	4	128	No Split	True	LYG	7.94×10^{2}	2.07×10^{-2}
	167	1	4	512	No Split	True	LYG	8.94×10^{2}	1.32×10^{-2}
	163	2	3	256	No Split	True	LYG	9.25×10^{2}	2.18×10^{-3}
	162	3	3	128	No Split	True	LYG	9.80×10^{2}	1.53×10^{-3}
	68	4	3	512	No Split	True	Both	9.91×10^{2}	2.13×10^{-2}
	166	5	4	256	No Split	True	LYG	1.04×10^{3}	3.60×10^{-2}
	117	6	4	128	Categories	True	Both	1.07×10^{3}	2.42×10^{-2}
	159	7	2	128	No Split	True	LYG	1.14×10^{3}	3.31×10^{-2}
	161	8	2	512	No Split	True	LYG	1.17×10^{3}	1.06×10^{-2}
	64	9	2	256	No Split	True	Both	1.23×10^{3}	4.60×10^{-2}
5: Orth, Orth	144	0	1	128	No Split	False	LYG	5.04×10^{3}	3.20×10^{-3}
	146	1	1	512	No Split	False	LYG	5.17×10^{3}	2.34×10^{-3}
	164	2	3	512	No Split	True	LYG	5.41×10^{3}	8.13×10^{-3}
	162	3	3	128	No Split	True	LYG	5.52×10^{3}	8.59×10^{-3}
	147	4	2	128	No Split	False	LYG	5.84×10^{3}	6.15×10^{-3}
	145	5	1	256	No Split	False	LYG	6.10×10^{3}	1.24×10^{-2}
	165	6	4	128	No Split	True	LYG	6.26×10^{3}	8.58×10^{-3}
	48	7	1	128	No Split	False	Both	6.62×10^{3}	9.71×10^{-3}
	148	8	2	256	No Split	False	LYG	6.66×10^{3}	3.13×10^{-3}
	149	9	2	512	No Split	False	LYG	6.92×10^{3}	3.38×10^{-3}

Table 17: Top 10 NN architectures for LYG prediction, for each sampling strategy.

No: Risk, Screen	ID	Rank	Layers	Nodes	Cost Split	BN	Y	MSE	SMAPE
1: Unif, Unif	85	0	1	256	No Split	True	Cost	1.28×10^{12}	1.01×10^{-3}
	84	1	1	128	No Split	True	Cost	1.32×10^{12}	2.55×10^{-3}
	95	2	4	512	No Split	True	Cost	1.66×10^{12}	1.01×10^{-3}
	86	3	1	512	No Split	True	Cost	1.70×10^{12}	1.16×10^{-2}
	120	4	1	128	Categories	False	Cost	1.82×10^{12}	3.07×10^{-3}
	121	5	1	256	Categories	False	Cost	1.95×10^{12}	6.78×10^{-3}
	89	6	2	512	No Split	True	Cost	1.96×10^{12}	2.42×10^{-3}
	77	7	2	512	No Split	False	Cost	2.00×10^{12}	2.46×10^{-3}
	67	8	3	256	No Split	True	Both	2.27×10^{12}	2.24×10^{-2}
	92	9	3	512	No Split	True	Cost	2.59×10^{12}	2.27×10^{-2}
2: Gam, Unif	87	0	2	128	No Split	True	Cost	1.80×10^{11}	3.96×10^{-3}
	90	1	3	128	No Split	True	Cost	2.06×10^{11}	4.54×10^{-3}
	85	2	1	256	No Split	True	Cost	2.20×10^{11}	1.21×10^{-3}
	84	3	1	128	No Split	True	Cost	2.23×10^{11}	3.06×10^{-3}
	89	4	2	512	No Split	True	Cost	2.43×10^{11}	5.26×10^{-3}
	86	5	1	512	No Split	True	Cost	2.70×10^{11}	3.59×10^{-3}
	88	6	2	256	No Split	True	Cost	2.87×10^{11}	5.78×10^{-3}
	95	7	4	512	No Split	True	Cost	2.93×10^{11}	6.32×10^{-4}
	94	8	4	256	No Split	True	Cost	2.95×10^{11}	2.28×10^{-3}
	91	9	3	256	No Split	True	Cost	4.73×10^{11}	1.25×10^{-2}
3: Unif, Freq	88	0	2	256	No Split	True	Cost	3.78×10^{12}	9.78×10^{-3}
	90	1	3	128	No Split	True	Cost	4.00×10^{12}	9.54×10^{-3}
	92	2	3	512	No Split	True	Cost	4.35×10^{12}	9.13×10^{-3}
	89	3	2	512	No Split	True	Cost	4.44×10^{12}	2.79×10^{-3}
	23	4	4	512	Full Split	True	Both	4.86×10^{12}	1.55×10^{-2}
	21	5	4	128	Full Split	True	Both	4.87×10^{12}	1.70×10^{-2}
	94	6	4	256	No Split	True	Cost	4.95×10^{12}	1.73×10^{-3}
	77	7	2	512	No Split	False	Cost	4.95×10^{12}	1.27×10^{-2}
	93	8	4	128	No Split	True	Cost	5.55×10^{12}	2.16×10^{-2}
	22	9	4	256	Full Split	True	Both	6.42×10^{12}	3.17×10^{-2}
4: Gam, Freq	93	0	4	128	No Split	True	Cost	1.31×10^{12}	3.00×10^{-2}
	75	1	2	128	No Split	False	Cost	1.32×10^{12}	5.73×10^{-2}
	95	2	4	512	No Split	True	Cost	1.42×10^{12}	2.80×10^{-2}
	73	3	1	256	No Split	False	Cost	1.95×10^{12}	1.77×10^{-2}
	76	4	2	256	No Split	False	Cost	2.13×10^{12}	1.58×10^{-2}
	74	5	1	512	No Split	False	Cost	2.22×10^{12}	2.60×10^{-2}
	89	6	2	512	No Split	True	Cost	2.39×10^{12}	2.75×10^{-2}
	142	7	4	256	Categories	True	Cost	2.41×10^{12}	5.86×10^{-2}
	91	8	3	256	No Split	True	Cost	2.46×10^{12}	2.12×10^{-2}
	94	9	4	256	No Split	True	Cost	2.71×10^{12}	7.65×10^{-3}
5: OLHS, OLHS	86	0	1	512	No Split	True	Cost	9.22×10^{11}	1.43×10^{-3}
	84	1	1	128	No Split	True	Cost	1.05×10^{12}	3.16×10^{-3}
	94	2	4	256	No Split	True	Cost	1.06×10^{12}	7.71×10^{-3}
	71	3	4	512	No Split	True	Both	1.07×10^{12}	1.40×10^{-2}
	85	4	1	256	No Split	True	Cost	1.10×10^{12}	3.54×10^{-3}
	70	5	4	256	No Split	True	Both	1.14×10^{12}	1.47×10^{-2}
	49	6	1	256	No Split	False	Both	1.14×10^{12}	1.32×10^{-2}
	61	7	1	256	No Split	True	Both	1.19×10^{12}	1.23×10^{-2}
	48	8	1	128	No Split	False	Both	1.20×10^{12}	1.26×10^{-2}
	60	9	1	128	No Split	True	Both	1.21×10^{12}	1.36×10^{-2}

Table 18: Top 10 NN architectures for Cost prediction, for each sampling strategy. The MSE and SMAPE scores provided are the sample average of predictions over five validation sets.

E.2 Experiment 2

	LYG				Cost			
	τ	lr	p	$loss$	τ	lr	p	$loss$
1	1.78×10^{-4}	1.00×10^{-4}	3.20×10^{-1}	6.22	1.77×10^{-12}	1.00×10^{-4}	2.26×10^{-1}	1.58×10^{1}
	1.37×10^{-4}	1.00×10^{-4}	3.12×10^{-1}	6.23	1.68×10^{-12}	1.00×10^{-4}	2.09×10^{-1}	1.58×10^{1}
	3.78×10^{-4}	1.00×10^{-4}	3.15×10^{-1}	6.23	6.98×10^{-13}	1.00×10^{-4}	2.41×10^{-1}	1.58×10^{1}
	3.28×10^{-4}	1.00×10^{-4}	3.04×10^{-1}	6.24	1.40×10^{-12}	1.00×10^{-4}	2.42×10^{-1}	1.58×10^{1}
	5.89×10^{-4}	1.00×10^{-4}	3.13×10^{-1}	6.26	1.42×10^{-12}	1.00×10^{-4}	2.01×10^{-1}	1.58×10^{1}
2	2.43×10^{-3}	1.00×10^{-2}	7.86×10^{-2}	4.73	6.42×10^{-10}	1.00×10^{-4}	1.63×10^{-1}	1.47×10^{1}
	1.77×10^{-3}	1.00×10^{-2}	1.23×10^{-1}	4.74	1.96×10^{-11}	1.00×10^{-4}	1.63×10^{-1}	1.48×10^{1}
	2.81×10^{-3}	1.00×10^{-3}	4.85×10^{-2}	4.75	1.27×10^{-10}	1.00×10^{-4}	1.49×10^{-1}	1.48×10^{1}
	4.06×10^{-3}	1.00×10^{-2}	7.90×10^{-2}	4.76	1.14×10^{-11}	1.00×10^{-4}	1.64×10^{-1}	1.48×10^{1}
	3.39×10^{-3}	1.00×10^{-4}	1.56×10^{-1}	4.76	2.23×10^{-10}	1.00×10^{-4}	1.37×10^{-1}	1.48×10^{1}
3	2.93×10^{-4}	1.00×10^{-3}	2.20×10^{-1}	5.92	6.08×10^{-11}	1.00×10^{-2}	3.51×10^{-1}	1.67×10^{1}
	3.09×10^{-4}	1.00×10^{-4}	2.25×10^{-1}	5.92	3.12×10^{-12}	1.00×10^{-2}	4.13×10^{-1}	1.67×10^{1}
	4.82×10^{-4}	1.00×10^{-4}	2.45×10^{-1}	5.93	3.90×10^{-11}	1.00×10^{-2}	3.13×10^{-1}	1.67×10^{1}
	5.01×10^{-4}	1.00×10^{-4}	2.47×10^{-1}	5.93	1.74×10^{-11}	1.00×10^{-2}	3.37×10^{-1}	1.67×10^{1}
	3.02×10^{-4}	1.00×10^{-3}	2.15×10^{-1}	5.94	6.11×10^{-11}	1.00×10^{-2}	3.57×10^{-1}	1.68×10^{1}
4	1.25×10^{-3}	1.00×10^{-3}	3.37×10^{-1}	4.91	3.97×10^{-11}	1.00×10^{-2}	2.02×10^{-1}	1.50×10^{1}
	6.21×10^{-3}	1.00×10^{-3}	2.67×10^{-1}	4.93	4.89×10^{-11}	1.00×10^{-2}	2.08×10^{-1}	1.51×10^{1}
	1.02×10^{-3}	1.00×10^{-3}	3.68×10^{-1}	4.96	1.04×10^{-10}	1.00×10^{-2}	2.04×10^{-1}	1.51×10^{1}
	5.36×10^{-4}	1.00×10^{-3}	1.88×10^{-1}	4.97	4.42×10^{-11}	1.00×10^{-2}	2.09×10^{-1}	1.51×10^{1}
	9.44×10^{-4}	1.00×10^{-3}	3.68×10^{-1}	4.98	1.03×10^{-11}	1.00×10^{-2}	2.10×10^{-1}	1.51×10^{1}
5	2.46×10^{-4}	1.00×10^{-2}	2.89×10^{-1}	5.95	4.90×10^{-11}	1.00×10^{-2}	4.04×10^{-1}	1.63×10^{1}
	3.72×10^{-4}	1.00×10^{-2}	2.77×10^{-1}	6.00	1.15×10^{-10}	1.00×10^{-2}	4.99×10^{-1}	1.65×10^{1}
	4.32×10^{-4}	1.00×10^{-2}	3.78×10^{-1}	6.07	8.80×10^{-11}	1.00×10^{-2}	4.02×10^{-1}	1.65×10^{1}
	4.90×10^{-4}	1.00×10^{-2}	3.87×10^{-1}	6.10	1.33×10^{-10}	1.00×10^{-2}	4.51×10^{-1}	1.66×10^{1}
	7.48×10^{-4}	1.00×10^{-2}	3.48×10^{-1}	6.15	9.67×10^{-11}	1.00×10^{-2}	4.32×10^{-1}	1.67×10^{1}

Table 19: Top 5 BNN hyperparameters, per sampling strategy

F Monte-Carlo Dropout

Following Bayes' theorem, we can formulate the posterior distribution of weights as

$$p(\theta|X,Y) = \frac{P(X,Y|\theta)P(\theta)}{P(X,Y)}. \tag{7}$$

The goal is to determine the parameters of this distribution, so we can predict some unobserved y from the holdout set y_h, based on the unobserved x_h, by integrating over the weights distribution. This yields

$$P(y_h|x_h,X,Y) = \int P(y_h|x_h,\theta)P(\theta|X,Y)d\theta, \tag{8}$$

which provides an exact answer, but is usually intractable to calculate. Recently, several authors have used variational schemes to approximate the optimal model parameters. One of the more notable, Kingma et al. (2015) use a sampling method to minimize the Kullback-Leibler Divergence (KL Divergence) (also known as relative entropy) divergence between $P(\theta|X,Y)$ and some approximating function $q(\theta|X,Y)$. The KL Divergence is then denoted by

$$KL(q(\mathbf{w}|\theta) \parallel P(\mathbf{w}|X,Y)) = \int q(\mathbf{w}|\theta) log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(X,Y|\mathbf{w})} d\mathbf{w}. \tag{9}$$

Even though these techniques offer vast efficiency improvements over exact methods, they are still computationally expensive and require a complex, entirely probabilistic Neural Network framework to execute. With an added uncertainty variable for all weights, this totals to double the amount of variables to estimate.

A popular technique of approximating an exact BNN is through Monte Carlo Dropout (MCD) (Gal & Ghahramani, 2016). An exact derivation can be found in the appendix of Gal and Ghahramani (2016) and is beyond the scope of this thesis. Their main contribution constitutes that keeping dropout layers turned on during inference time, will yield a mean and standard deviation that approximate the posterior mean and standard deviation of a BNN. Instrumental towards this conclusion is their derivation that the loss function of an ANN with L layers and softmax loss or MSE loss, with L_2 regularization, denoted by

$$\Lambda_{dropout} = \frac{1}{N}\sum_{i=1}^{N} E(y_i,\hat{y}_i) + \lambda \sum_{i=1}^{L}(\|\mathbf{W}_i\|_2^2 + \|\mathbf{b}_i\|_2^2), \tag{10}$$

where λ is the weight decay, a parameter for L_2 regularization. The dropout loss in equation 10 is approximately the same as the loss of a Deep Gaussian Process (DGP) (Damianou & Lawrence, 2013), which is an important result because it implies a Gaussian distribution for its outcomes; this applies only when a trick similar to equation 9 is used, where the posterior distribution of the model weights is approximated by a distribution of the weight matrices in which the the columns are randomly set to zero through some Bernoulli process. The latter is analogous to dropout, completing the circle.

They further show this can be used to derive expectation and uncertainty for the model by drawing Monte Carlo samples from this Bernoulli Distribution, obtaining the first two

moments, and inferring the mean,

$$E_{\mathbf{y}_h|\mathbf{x}_h}(\mathbf{y}_h) \approx \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{y}}_h(\mathbf{x}_h, \mathbf{W}_1^t, ..., \mathbf{W}_L^t), \tag{11}$$

and the variance, namely,

$$\begin{aligned} Var_{\mathbf{y}_h|\mathbf{x}_h}(\mathbf{y}_h) \approx \quad & \tau^{-1}\mathbf{I}_D \\ & + \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{y}}_h(\mathbf{x}_h, \mathbf{W}_1^t, ..., \mathbf{W}_L^t)^{tr}\hat{\mathbf{y}}_h(\mathbf{x}_h, \mathbf{W}_1^t, ..., \mathbf{W}_L^t) \\ & - E_{\mathbf{y}_h|\mathbf{x}_h}(\mathbf{y}_h)^{tr}E_{\mathbf{y}_h|\mathbf{x}_h}(\mathbf{y}_h). \end{aligned} \tag{12}$$

Here, τ is the model precision, which is a function of some known model parameters, through

$$\tau = \frac{pl^2}{2N\lambda}, \tag{13}$$

where l is the prior length-scale, N is the number of observations and p and λ are as before (Gal & Ghahramani, 2016). The prior length-scale is used to determine a prior initialization of the weights of the first layer, W_1. A larger length-scale means means stronger regularization over the weights, a smaller length-scale leads to higher precision. Equations 11 and 12 equal the sample mean and sample variance, respectively, of T forward passes through the ANN with Dropout turned on.

In summary, prediction mean and variance may be obtained by using dropout layers for predictions. Intuitively, this means that an ANN, trained with dropout layers to convergence, may be used to predict y_t a high amount of times (e.g. 10,000) on a single test observation x_t. Due to the probabilistic nature of the dropout layers, this will yield a stochastic sample of estimates over which we can calculate the sample mean and sample variance. The sample variance is an estimate of the model's epistemic uncertainty. To obtain the total prediction uncertainty, we add the aleatoric uncertainty, which is obtained by taking the inverse of the model precision.

References

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., ... Zhu, Z. (2016, 20–22 Jun). Deep speech 2 : End-to-end speech recognition in english and mandarin. In M. F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd international conference on machine learning* (Vol. 48, pp. 173–182). New York, New York, USA: PMLR. Retrieved from http://proceedings.mlr.press/v48/amodei16.html

Baldi, P., & Sadowski, P. J. (2013). Understanding dropout. In *Advances in neural information processing systems* (pp. 2814–2822).

Bergstra, J., Yamins, D., & Cox, D. D. (n.d.). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *To appear in proc. of the 30th international conference on machine learning (icml 2013)*.

Bouhlel, M. A. i., Bartoli, N., Otsmane, A., & Morlier, J. (2016). Improving kriging surrogates of high-dimensional design models by partial least squares dimension reduction. *Structural and Multidisciplinary Optimization*, *53*(5), 935–952.

Bray, F., Ferlay, J., Soerjomataram, I., Siegel, R. L., Torre, L. A., & Jemal, A. (2018). Global cancer statistics 2018: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA: a cancer journal for clinicians*, *68*(6), 394–424.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Caruana, R., Lawrence, S., & Giles, C. L. (2001). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems* (pp. 402–408).

Cenin, D. R., St John, D. J. B., Ledger, M. J. N., Slevin, T., & Lansdorp-Vogelaar, I. (2014). Optimising the expansion of the national bowel cancer screening program. *Medical Journal of Australia*, *201*(8), 456–461.

Cobb, A. D., Himes, M. D., Soboczenski, F., Zorzan, S., O'Beirne, M. D., Baydin, A. G., ... others (2019). An ensemble of bayesian neural networks for exoplanetary atmospheric retrieval. *The Astronomical Journal*, *158*(1), 33.

Damianou, A., & Lawrence, N. (2013). Deep gaussian processes. In *Artificial intelligence and statistics* (pp. 207–215).

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*(Jul), 2121–2159.

Fox, C. R., & Ülkümen, G. (2011). Distinguishing two dimensions of uncertainty. *Perspectives on thinking, judging, and decision making*, 21–35.

Freedman, D. A. (2009). *Statistical models: theory and practice*. Cambridge University Press.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (pp. 1050–1059).

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, *4*(1), 1–58.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 315–323).

Goede, S. L., Rabeneck, L., Lansdorp-Vogelaar, I., Zauber, A. G., Paszat, L. F., Hoch, J. S., ... Van Ballegooijen, M. (2015). The impact of stratifying by family history in colorectal cancer screening programs. *International journal of cancer*, *137*(5), 1119–1127.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Haggar, F. A., & Boushey, R. P. (2009). Colorectal cancer epidemiology: incidence, mortality, survival, and risk factors. *Clinics in colon and rectal surgery*, *22*(04), 191–197.

Hastie, T., Tibshirani, R., Friedman, J., & Franklin, J. (2005). The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, *27*(2), 83–85.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Hol, L., Van Leerdam, M. E., Van Ballegooijen, M., Van Vuuren, A. J., Van Dekken, H., Reijerink, J. C., ... Kuipers, E. J. (2010). Screening for colorectal cancer: randomised trial comparing guaiac-based and immunochemical faecal occult blood testing and flexible sigmoidoscopy. *Gut*, *59*(01), 62–68.

Hussain, M. F., Barton, R. R., & Joshi, S. B. (2002). Metamodeling: radial basis functions, versus polynomials. *European Journal of Operational Research*, *138*(1), 142–154.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jemal, A., Clegg, L. X., Ward, E., Ries, L. A., Wu, X., Jamison, P. M., ... Edwards, B. K. (2004). Annual report to the nation on the status of cancer, 1975–2001, with a special feature regarding survival. *Cancer: Interdisciplinary International Journal of the American Cancer Society*, *101*(1), 3–27.

Johnson, C. M., Wei, C., Ensor, J. E., Smolenski, D. J., Amos, C. I., Levin, B., & Berry, D. A. (2013). Meta-analyses of colorectal cancer risk factors. *Cancer causes & control*, *24*(6), 1207–1222.

Kaminski, M. F., Regula, J., Kraszewska, E., Polkowski, M., Wojciechowska, U., Didkowska, J., ... Butruk, E. (2010). Quality indicators for colonoscopy and the risk of interval cancer. *New England Journal of Medicine*, *362*(19), 1795–1803.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., Salimans, T., & Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems* (pp. 2575–2583).

Kleijnen, J. P., & Sargent, R. G. (2000). A methodology for fitting and validating metamodels in simulation. *European Journal of Operational Research*, *120*(1), 14–29.

Knudsen, A. B., Zauber, A. G., Rutter, C. M., Naber, S. K., Doria-Rose, V. P., Pabiniak, C., ... Kuntz, K. M. (2016). Estimation of benefits, burden, and harms of colorectal cancer screening strategies: modeling study for the us preventive services task force. *Jama*, *315*(23), 2595–2609.

Kolassa, S., & Martin, R. (2011). Percentage errors can ruin your day (and rolling the dice shows how). *Foresight: The International Journal of Applied Forecasting*(23).

Lai, K. K., Yu, L., Huang, W., & Wang, S. (2006). A novel support vector machine metamodel for business risk identification. In *Pacific rim international conference on artificial intelligence* (pp. 980–984).

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems* (pp. 6402–6413).

Lauby-Secretan, B., Vilahur, N., Bianchini, F., Guha, N., & Straif, K. (2018). The iarc perspective on colorectal cancer screening. *New England Journal of Medicine*, *378*(18), 1734–1740.

Loeve, F., Boer, R., van Oortmarssen, G. J., van Ballegooijen, M., & Habbema, J. D. F. (1999). The miscan-colon simulation model for the evaluation of colorectal cancer screening. *Computers and Biomedical Research*, *32*(1), 13–33.

Lu, L., Shin, Y., Su, Y., & Karniadakis, G. E. (2019). Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*.

Ma, G. K., & Ladabaum, U. (2014). Personalizing colorectal cancer screening: a systematic review of models to predict risk of colorectal neoplasia. *Clinical Gastroenterology and Hepatology*, *12*(10), 1624–1634.

Mentch, L., & Hooker, G. (2016). Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *The Journal of Machine Learning Research*, *17*(1), 841–881.

Morris, M. D., & Mitchell, T. J. (1995). Exploratory designs for computational experiments. *Journal of statistical planning and inference*, *43*(3), 381–402.

Morson, B. (1974). The polyp-cancer sequence in the large bowel. *Proc R Soc Med*, *67*(6 Pt 1), 451-457.

Naber, S. (2017). *Reducing harms and increasing benefits of screening for cervical cancer and colorectal cancer – a model-based approach* (Doctoral dissertation, E). Retrieved from http://hdl.handle.net/1765/98677

Neal, R. M. (2012). *Bayesian learning for neural networks* (Vol. 118). Springer Science & Business Media.

Peterse, E. F., Meester, R. G., Siegel, R. L., Chen, J. C., Dwyer, A., Ahnen, D. J., ... Lansdorp-Vogelaar, I. (2018). The impact of the rising colorectal cancer incidence in young adults on the optimal age to start screening: microsimulation analysis i to inform the american cancer society colorectal cancer screening guideline. *Cancer*, *124*(14), 2964–2973.

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Ries, L. A., Harkins, D., Krapcho, M., Mariotto, A., Miller, B., Feuer, E. J., ... others (2006). Seer cancer statistics review, 1975-2003.

Ristvedt, S. L., Birnbaum, E. H., Dietz, D. W., Fleshman, J. W., Kodner, I. J., & Read, T. E. (2005). Delayed treatment for rectal cancer. *Diseases of the colon & rectum*, *48*(9), 1736–1741.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Roßbach, P. (2018, Oct). *https://blog.frankfurt-school.de.* Frankfurt School of Finance & Management. Retrieved from https://blog.frankfurt-school.de/wp-content/uploads/2018/10/Neural-Networks-vs-Random-Forests.pdf

Rutter, C. M., Knudsen, A. B., Marsh, T. L., Doria-Rose, V. P., Johnson, E., Pabiniak, C., ... Lansdorp-Vogelaar, I. (2016). Validation of models used to inform colorectal cancer screening guidelines: accuracy and implications. *Medical Decision Making*, *36*(5), 604–614.

Sargent, R. G. (2010). Verification and validation of simulation models. In *Proceedings of the 2010 winter simulation conference* (pp. 166–183).

Scalia, G., Grambow, C. A., Pernici, B., Li, Y.-P., & Green, W. H. (2019). Evaluating scalable uncertainty estimation methods for dnn-based molecular property prediction. *arXiv preprint arXiv:1910.03127*.

Scotto Di Perrotolo, A. (2018). *A theoretical framework for bayesian optimization convergence* (Doctoral dissertation). Retrieved from https://pdfs.semanticscholar.org/7521/226fe6b7f01fc1e37c5cb167fee38a08d060.pdf

Segnan, N., Senore, C., Andreoni, B., Azzoni, A., Bisanti, L., Cardelli, A., ... others (2007). Comparing attendance and detection rate of colonoscopy with sigmoidoscopy and fit for colorectal cancer screening. *Gastroenterology*, *132*(7), 2304–2312.

Siegel, R. L., Fedewa, S. A., Anderson, W. F., Miller, K. D., Ma, J., Rosenberg, P. S., & Jemal, A. (2017). Colorectal cancer incidence patterns in the united states, 1974–2013. *JNCI: Journal of the National Cancer Institute*, *109*(8).

Singh, H., Turner, D., Xue, L., Targownik, L. E., & Bernstein, C. N. (2006). Risk of developing colorectal cancer following a negative colonoscopy examination: evidence for a 10-year interval between colonoscopies. *Jama*, *295*(20), 2366–2373.

Sóbester, A., Forrester, A. I., Toal, D. J., Tresidder, E., & Tucker, S. (2014). Engineering design applications of surrogate-assisted optimization techniques. *Optimization and Engineering*, *15*(1), 243–265.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Syberfeldt, A., Grimm, H., & Ng, A. (2008). Design of experiments for training metamodels in simulation-based optimisation of manufacturing systems. In *The 18th international conference on flexible automation and intelligent manufacturing (faim'08) skövde,*

sweden, june 30-july 2, 2008.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016, June). Rethinking the inception architecture for computer vision. In *The ieee conference on computer vision and pattern recognition (cvpr).*

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, *4*(2), 26–31.

Toes-Zoutendijk, E., van Leerdam, M. E., Dekker, E., van Hees, F., Penning, C., Nagtegaal, I., ... others (2017). Real-time monitoring of results during first year of dutch colorectal cancer screening program and optimization by altering fecal immunochemical test cut-off levels. *Gastroenterology*, *152*(4), 767–775.

Usher-Smith, J., Harshfield, A., Saunders, C., Sharp, S., Emery, J., Walter, F., ... Griffin, S. (2018). External validation of risk prediction models for incident colorectal cancer using uk biobank. *British journal of cancer*, *118*(5), 750.

Van Hees, F., Zauber, A. G., van Veldhuizen, H., Heijnen, M.-L. A., Penning, C., de Koning, H. J., ... Lansdorp-Vogelaar, I. (2015). The value of models in informing resource allocation in colorectal cancer screening: the case of the netherlands. *Gut*, *64*(12), 1985–1997.

Van Rijn, J. C., Reitsma, J. B., Stoker, J., Bossuyt, P. M., Van Deventer, S. J., & Dekker, E. (2006). Polyp miss rate determined by tandem colonoscopy: a systematic review. *The American journal of gastroenterology*, *101*(2), 343.

Van Rossum, L. G., Van Rijn, A. F., Laheij, R. J., Van Oijen, M. G., Fockens, P., Van Krieken, H. H., ... Dekker, E. (2008). Random comparison of guaiac and immunochemical fecal occult blood tests for colorectal cancer in a screening population. *Gastroenterology*, *135*(1), 82–90.

Villa-Vialaneix, N., Follador, M., Ratto, M., & Leip, A. (2012). A comparison of eight metamodeling techniques for the simulation of n2o fluxes and n leaching from corn crops. *Environmental Modelling & Software*, *34*, 51–66.

Vogelstein, B., Fearon, E. R., Hamilton, S. R., Kern, S. E., Preisinger, A. C., Leppert, M., ... Bos, J. L. (1988). Genetic alterations during colorectal-tumor development. *New England Journal of Medicine*, *319*(9), 525–532.

Wolpert, D. H., Macready, W. G., et al. (1995). *No free lunch theorems for search* (Tech. Rep.). Technical Report SFI-TR-95-02-010, Santa Fe Institute.

Wolpert, D. H., Macready, W. G., et al. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67–82.

Ykema, B., Rigter, L., Spaander, M., Moons, L., Bisseling, T., Aleman, B., ... others (2020). Diagnostic accuracy of stool tests for colorectal cancer surveillance in hodgkin lymphoma survivors. *Journal of Clinical Medicine*, *9*(1), 190.

Zhu, L., & Laptev, N. (2017). Deep and confident prediction for time series at uber. In *2017 ieee international conference on data mining workshops (icdmw)* (pp. 103–110).